UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

VISUALIZAÇÃO DE MÚSICA À DISTÂNCIA DE UM GESTO

João Tiago Sobral Gomes

PROJETO

MESTRADO EM INFORMÁTICA

2014

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

VISUALIZAÇÃO DE MÚSICA À DISTÂNCIA DE UM GESTO

João Tiago Sobral Gomes

PROJETO

MESTRADO EM INFORMÁTICA

Trabalho orientado pela Profª. Doutora Maria Beatriz Duarte Pereira do Carmo
e co-orientado pela Profª Doutora Ana Paula Boler Cláudio

2014

Agradecimentos

Em primeiro lugar, gostava de agradecer à Professora Ana Paula Cláudio e à Professora Beatriz Carmo, que orientaram este projeto durante um ano e que me encorajaram desde o início a criar esta aplicação quando ela ainda não passava de uma ideia. Por intermédio das minhas orientadoras queria também agradecer à Faculdade de Ciências, em particular ao Departamento de Informática, e ainda à Fundação para a Ciência e Tecnologia e ao LabMAg pelo apoio financeiro, consubstanciado numa Bolsa Pest OE(EEI/UI0434/2011) com a duração de 4 meses.

Devo ao Paulo Baeta o acompanhamento do projeto desde o primeiro momento e a disponibilidade para todas as dúvidas de última hora, principalmente na vertente musical. Sem ele este projeto seria algo muito diferente.

Gostava de agradecer também ao Sérgio Serra, que desenvolveu o seu projeto ao mesmo tempo que eu e fez este ano passar com muito mais facilidade.

Obrigado ao André Santos e à Ânia Finuras por terem tido a paciência de me aturar durante este ano.

À minha mãe e irmã

Resumo

Este projeto consistiu na construção de uma aplicação *web* de cariz musical que permite aos utilizadores tocar e compor vários sons. Estes sons são sobrepostos interativamente e em tempo-real pelo utilizador, de modo a criar música com crescente complexidade. Para tocar os sons de forma natural é usado o dispositivo *Leap Motion*, capaz de medir a posição tridimensional das mãos do utilizador. A interação é acompanhada de uma visualização tridimensional que permite, não só servir de guia ao utilizador na sua composição, como também visualizar a música que está a ser criada.

O carácter inovador da aplicação reside no facto de se juntar numa só aplicação as componentes musical, de interação com o dispositivo *Leap Motion* e a visualização da música criada tudo num ambiente em tempo-real. Para isso foi investigado o processo de criação e composição de música utilizado por certos artistas contemporâneos. Tentou-se transpor esse processo para a aplicação criada, ainda que a uma escala mais pequena para facilitar o processo criativo do utilizador comum. Finalmente, a aplicação foi testada com vários utilizadores para avaliar algumas características da sua usabilidade.

Palavras-chave: visualização, música, composição musical, leap motion, interação natural

Abstract

In this project a web application of musical nature was developed allowing users to play and compose various sounds. These sounds are overlapped interactively and in real-time by the user to create music of increased complexity. To play the sounds in a natural fashion the Leap Motion device is used, it is capable of tracking the users 3D hand position. The interaction is accompanied with a 3D visualization that servers two purposes, to serve as a guide to the user's composition and to visualize the music that is being created.

The innovative nature of this application lies in the combination of the musical component, the interaction with the Leap Motion device and the music visualization, all in real-time. The musical creation and composition process used by contemporary artists was studied. That process, although on a smaller scale, was transferred to the application in order to facilitate the creative process of the common user. Finally, the application was tested with multiple users to assess some usability characteristics of the final application.

Keywords: visualization, music, musical composition, leap motion, natural interaction

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este relatório descreve o trabalho desenvolvido no Laboratório de Investigação LabMAg no âmbito do Projeto em Informática do Mestrado em Informática do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa com o título *Visualização de Música à Distância de um Gesto*.

Apresentam-se neste capítulo os principais temas abordados neste projeto e que o motivaram bem como os objetivos principais e a organização do documento.

1.1 Motivação

Lev Sergeevich Termen patenteou em 1928 um instrumento musical electrónico com o nome *Theremin* (Ssergejewitsch, 1928), que era controlado pelo músico sem qualquer contacto físico. O instrumento dispõe de duas antenas que controlam respetivamente o *pitch* (ou frequência) e a amplitude (ou volume) do som. O músico controla cada antena com cada mão, afastando ou aproximando as mãos das antenas de modo a fazer variar o som do instrumento (Pringle, 2009).

A 21 de Maio de 2012 foi anunciado pela empresa *Leap Motion* (Leap Motion) um dispositivo, com o mesmo nome, constituído por duas câmaras de raios infravermelhos e três *LED* infravermelhos, as quais permitem fazer o *tracking* dos dedos e mãos de um utilizador num espaço mais ou menos hemisférico com cerca de 1 metro de alcance (Leap Motion Structured Light Pattern, 2013).

As semelhanças entre um *Theremin* e um dispositivo *Leap Motion* são óbvias. Daí surgiu a ideia de usar o *Leap Motion* para, não só tocar um instrumento virtual, tal como acontece com o *Theremin*, como também para servir de ferramenta de composição

musical e para criar músicas com vários sons que, para esse efeito, se sobrepõem uns aos outros.

Com o evoluir da tecnologia musical, é hoje possível a apenas uma pessoa não só compor como também tocar vários instrumentos sozinha, e sobrepor os vários sons de forma a fazê-los soar como uma banda ou uma orquestra inteira. Existem já vários artistas que fazem isso mesmo. Estes artistas não estão ligados a um só género musical. Existem artistas que seguem géneros que vão desde o *Hip Hop*, passando pelo *Rap, Pop*, *Soul, Jazz, RnB*, e até música Erudita. Serão apresentados alguns deles no próximo capítulo.

Quanto à visualização musical, existem hoje em dia muitas formas de visualização de música, no entanto a mais proeminente de todas é a notação musical que é usada já há vários séculos. Este tipo de visualização musical, todavia, só pode ser compreendida por alguns que, para o fazer, tiveram formação musical.

Estas três vertentes − composição/reprodução musical, interação com um dispositivo que sirva de instrumento virtual e, finalmente, a visualização musical – serviram de inspiração para o desenvolvimento de uma aplicação que as combina.

1.2 Objetivos

O objetivo do trabalho foi integrar três componentes: composição/reprodução musical, interação com o dispositivo *Leap Motion* e visualização musical. Desenvolveu-se uma aplicação interativa com uma interface natural para criar música. O propósito não foi construir uma aplicação muito completa e com múltiplas funcionalidades, pois os limites temporais deste trabalho não o permitiam mas apenas criar um protótipo com um conjunto reduzido de funcionalidades, que sirva de prova de conceito. O que se propõe, é que seja uma aplicação de natureza essencialmente artística e que permita ao utilizador explorar a composição musical.

Expostas as principais áreas em que este projeto se insere, pretende-se, em primeiro lugar, explorar vários modos de interação de um utilizador com o dispositivo *Leap Motion*, de modo a que a composição ou modificação de sons pareça natural. Para isso é preciso ter em conta os vários graus de liberdade que este dispositivo nos proporciona, bem como a sua pertinência.

Em segundo lugar está a composição musical em si mesma e a exploração da melhor maneira de reproduzir os vários sons que estão à disposição do utilizador. Aqui é importante ter em conta a liberdade artística oferecida ao utilizador.

Por último, no que concerne à parte gráfica, foi dado um grande ênfase ao *feedback* visual do que está a acontecer. Isto toma essencialmente duas formas, ainda que obviamente relacionadas. Por um lado, temos a interface e a forma como o utilizador interage visualmente com a aplicação e a forma como esta interface se "liga", tanto com o dispositivo *Leap Motion*, como com a parte musical descrita anteriormente. Por outro lado, temos a visualização propriamente da música que está a ser criada ou manipulada naquele momento.

1.3 Estrutura do Documento

Este documento está organizado da seguinte forma:

- **Capítulo 2 – Trabalho relacionado**

  Neste capítulo são apresentados vários artistas contemporâneos que usam um processo específico para tocar e cantar parecendo que são várias pessoas ou instrumentos. São analisadas várias aplicações que usam tanto gráficos em tempo real como gráficos pré-processados para a visualização de música e finalmente são analisadas aplicações que servem não só para a visualização de música mas também para a sua criação.

- **Capítulo 3 – Trabalho realizado**

  Neste capítulo são discutidas as várias componentes que constituem a aplicação desenvolvida:

  - **Áudio** – explica-se a marcação de tempo, as várias *tracks* e sons que compõe a aplicação e a gravação e reprodução de som em tempo real. Toda esta componente teve que ser desenvolvida tendo em conta o desempenho devido à natureza em tempo-real que a aplicação possui.
  - **Interação para a reprodução de sons –** descreve-se a interação do utilizador com o dispositivo *Leap Motion* e as estratégias seguidas para que esta interação fosse o mais natural possível.

- **Interface para a construção, composição de múscia e marcação de tempo –** apresenta-se como é que a interface a que o utilizador tem acesso o ajuda a saber o estado das várias *tracks* e do que está a ser tocado ou reproduzido.
- **Visualização –** explica-se a visualização tridimensional da música que está a ser tocada e reproduzida

- **Capítulo 4 – Testes com utilizadores**

  Neste capítulo são apresentados os testes que foram feitos com utilizadores; mais especificamente, utilizadores comuns e peritos nas áreas musicais e de interação, para recolher informação acerca da usabilidade da aplicação.

- **Capítulo 5 – Conclusões e trabalho futuro**

  Neste capítulo são discutidas as conclusões tiradas do trabalho desenvolvido e são apresentadas sugestões para melhoramentos futuros.

Capítulo 2

Trabalho relacionado

Neste capítulo apresenta-se uma análise sobre os trabalhos desenvolvidos na área da composição musical realizado por vários artistas e é feito um levantamento relativo à visualização de música, em particular para a *web*. A apresentação dos trabalhos relativos à visualização de música encontra-se dividido em três áreas distintas: visualizações feitas em tempo real, visualizações pré-processadas e visualizações que permitem ao mesmo tempo criar música.

2.1 Artistas Musicais

Esta secção pretende dar várias referências de artistas musicais que usam a voz ou instrumentos para criarem músicas. A criação de música é feita somente pelo artista e segue um processo que é mais ou menos comum a todos os artistas referidos. Salvo pequenas variações, o processo consiste no seguinte: em primeiro lugar o artista dispõe sempre de equipamento (*hardware*) e/ou tecnologia (*software*) que lhe permite gravar várias *tracks* (em português, faixas, mas é importante não confundir com o termo faixas que é usado coloquialmente para nos referirmos a uma música de um álbum). Primeiramente o artista começa por gravar numa *track* um som, quer seja ele vindo da sua própria voz quer seja de um instrumento. Após a gravação, esse som começa a ser reproduzido imediatamente em *loop*. De seguida é gravado outro som noutra *track,* enquanto o anterior ainda está a ser reproduzido. Este processo vai-se repetindo, de forma a que as várias *tracks* se vão sobrepondo umas às outras para formar uma música cada vez mais complexa.

A música produzida com este processo pode ser bastante complexa, com várias *tracks* sobrepostas e a reproduzirem vários sons, ou algo simples com poucas *tracks*:

cabe ao artista decidir. Acrescento que a música produzida por este processo não tem que ser obrigatoriamente linear, no sentido em que vamos adicionando cada vez mais *tracks* que contêm vários sons para formar a música; uma música pode ter várias partes, como um refrão ou um solo. Assim, é possível gravar um conjunto de *tracks* que componham o refrão ou o solo e reproduzi-las em diferentes alturas.

Finalmente, o grau de automatização deste processo difere de artista para artista, o que se deve em parte ao nível de controlo que o artista quer ter sobre a música que produz e em parte ao tipo de música.

2.1.1 Jarle Bernhoft – Soul

Jarle Bernhoft (figura 1) é um cantor e compositor norueguês alternativo que fez parte da popular banda norueguesa *Span* (Bernhoft). Após a banda ter-se dividido em 2005, Jarle embarcou numa carreira a solo, tocando sozinho vários instrumentos e cantando principalmente temas Soul. Para além de ter já lançado vários discos, quer com a sua antiga banda *Span,* quer a solo, Jarle é também bastante conhecido no *Youtube*, tendo o seu vídeo mais popular acima de 3.000.000 de visualizações. Para além de tocar guitarra e piano, entre outros instrumentos, Jarle usa bastante a sua voz para criar sons, maioritariamente de percussão, ao estilo *beatbox*. Jarle recorre bastante a equipamento musical que lhe permite gravar uma *track* de som enquanto a anterior continua a ser reproduzida, bem como ligar e desligar *tracks* em tempo real. Isto é bastante notório nos seus vídeos do Youtube (Jarle Bernhoft Artist Biography by Jason Birchmeier).

Figura 1 - Jarle Bernhoft a intrepertar a canção C'mon Talk (Bernhoft - C'mon Talk (Official Video), 2011)

Segundo o próprio Jarle Bernhoft, foi por causa da crise económica e financeira de 2008 que este artista começou a compor sozinho e com recurso a esta tecnologia. Jarle tinha escrito um album inteiro de canções que envolviam grandes arranjos musicais e muitos instrumentos e pessoas, mas por não ter possibilidades financeiras para gravar o album desta maneira, decidiu alterar as músicas que já tinha escrito de forma a que ele mesmo as pudesse interpretar inteiramente sozinho e com recurso a tecnologia.

Isto serve para mostrar como alguém que consiga cantar e tocar vários intrumentos pode utilizar esta tecnologia e processo para, por exemplo, compor uma música primeiro sozinho e posteriormente juntar-se a músicos, tocando cada um deles as partes que anteriormente eram todas feitas pela mesma pessoa. Pode portanto ser uma maneira de mostrar a outras pessoas o que é que alguém tem em mente, em vez de o expressar por palavras, como é habitual.

2.1.2 Tom Thum – Jazz

Tom Tum (figura 2) é um artista e *stand up comedian* australiano que usa unicamente a voz, tanto para compor canções como para fazer espetáculos de comédia. É conhecido por fazer sons de *beatbox* mas também por fazer músicas de *Jazz* como se pode comprovar no seguinte vídeo: *Beatbox brilliance: Tom Thum at TEDxSydney* (Beatbox brilliance: Tom Thum at TEDxSydney, 2013). Sempre com um grande sentido de humor, Tom Thum usa neste vídeo a sua voz e, usando o processo descrito anteriormente, faz uma música de *Jazz*.

Figura 2 - Tom Thum a construir uma música ao vivo *(Beatbox brilliance: Tom Thum at TEDxSydney)*

Desconstruindo um pouco o que acontece no vídeo, Tom começa pela percussão e o primeiro som que faz é o de um prato de choques duma bateria, seguido logo por um som equivalente a um instrumento de percussão que consiste numa bola com algum tipo de areia lá dentro. Faz isto para definir uma linha base que serve como batida da música. Seguindo ainda a linha base da música, de seguida imita o som de um contrabaixo. Com a parte da percussão e a base feita passa para o trompete. Com o som do trompete não faz apenas um, mas dois sons cantados com afinações diferentes, de modo a não parecer um só instrumento. Depois volta de novo à parte da bateria e simula os sons do bombo e da tarola. Chegando a este ponto, está criada a harmonia e o ritmo base da música. Simula então o som do trompete mais uma vez, mas agora para fazer um solo em vez de o usar para as harmonias como anteriormente. Recorre novamente à bateria mas desta vez para um solo em vez de servir como percussão base, como acima descrito. Temos, por fim, o trompete com um pouco de improvisação.

Analisando o vídeo, percebemos como é que os vários instrumentos simulados são usados para criar uma pequena música de Jazz. Ficamos com a noção básica das três partes que compõem a música: ritmo, que é constituído pelos sons de percussão, bateria e contrabaixo, harmonias que são dadas pelos trompetes e finalmente melodias que são dadas pelo som do trompete também. Este vídeo é o mais interessante deste artista para vermos de que modo é que se pode ir construindo uma música aos poucos, sendo para mais esta música de Jazz, que é um estilo que prima pela complexidade e pela improvisação.

2.1.3 DubFx - Hip hop

Benjamin Stanford, também conhecido como Dub FX (figura 3), é um artista de rua originalmente de St. Kilda, Melbourne, Austrália. Dub FX ganhou bastante popularidade nos seus vídeos do Youtube, que contam com várias dezenas de milhões de visualizações (DUBFX) (Dubbing into Dub FX) e onde podemos vê-lo a interpretar vários dos seus temas quase sempre em ambiente de rua. O estilo musical de DubFx é uma mistura de *hip hop, reggae* e *drum n'bass*.

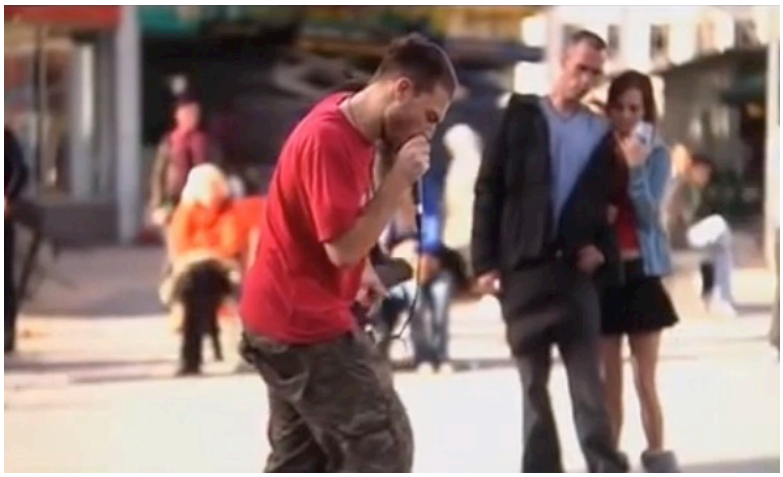


Figure 3 - DubFX a intrepertar o tema Love Someone em público, na rua (DubFX - Love Someone)

Num dos seus vídeos, *Dub FX 10/10/2008 'Love Someone'* (Dub FX 10/10/2008 'Love Someone', 2008), ele explica-nos como é que faz as várias partes da música e a vai construindo à medida que vai explicando o que é que está a fazer com o equipamento musical. Dub FX recorre bastante a pedais, que produzem variados efeitos, de modo a mudar a sua voz e poder imitar vários instrumentos.

2.1.4 Zoë Keating – Erudita

Zoë Keating é uma violoncelista e compositora nascida em Guelph, em Ontario, no Canadá e atualmente residente em São Francisco, na Califórnia. Informática de profissão, Zoë começou a ter aulas de violoncelo e a receber formação em música clássica desde os 8 anos de idade. Fez parte de várias bandas e projetos, desde bandas sonoras para televisão e para filmes a várias colaborações com vários artistas e bandas (Bio).

Apesar das suas colaborações com outros artistas, o mais interessante acerca de Zoë é a sua carreira a solo. Podemos ver no vídeo *Zoe Keating live - 'Lost' [HD] Sound Quality, ABC Radio National* (Zoe Keating live - 'Lost' [HD] Sound Quality, ABC Radio National, 2012) como consegue fazer inteiramente sozinha (mas com recurso a tecnologia, claro) uma música que soa como se fosse uma orquestra inteira a tocar. Para isso ela usa várias técnicas, *software* e *hardware*. Mais uma vez, o processo utilizado de sobrepor *tracks* é basicamente o mesmo que já foi explicado anteriormente.

Figura 4 - Zoë Keating a intrepertar a música Lost em estúdio. Dispõe de um portátil ao seu lado esquerdo e de um tapete do lado direito que não se encontra na imagem (Zoe Keating - Lost)

Assim, para além do violoncelo e do respetivo arco, que usa para tocar, Zoë tem essencialmente um computador ao seu lado, bem como um controlador a seus pés (figura 4). No computador é utilizado *software*, nomeadamente *Ableton Live* (Ableton) (que é bastante utilizado hoje em dia para *performances* ao vivo) e *SooperLooper* (Sooperlooper), que é um *software* que permite fazer *multi-sample looping* ao vivo (Sooperlooper). Ao lado do seu pé direito tem um *Keith McMillen Instruments SoftStep Foot Controller* (SOFTSTEP), que usa para gravar e reproduzir os vários excertos que toca. O processo utilizado por Zoë é um pouco diferente do dos outros artistas referidos anteriormente, pois neste caso as músicas por ela tocadas não são tão pautadas pela improvisação como as dos artistas anteriores. Isto não só porque as músicas já foram compostas anteriormente, o que não invalidaria a improvisação, mas por causa da natureza erudita da música, que deixa pouco ou nenhum espaço para o improviso.

Outra razão pela qual o seu processo é diferente do de outros artista é a automatização. Como já foi referido anteriormente, isto prende-se maioritariamente com o facto de estarmos a lidar com música erudita. Assim, e tal como Zoë explica numa entrevista que concedeu, ela tem que preparar o *software* e *hardware* para a música que vai tocar, sendo que para cada música esta preparação é diferente. Parte da automatização consiste, por exemplo, em programar o *software* de modo a que ela possa gravar durante um determinado número de compassos e reproduzir esses compassos também durante um determinado número de vezes. Ainda assim, nem todo o processo está automatizado: para a parte manual Zoë usa o tapete junto do seu pé direito para

controlar as pistas de áudio que está a reproduzir ou gravar. Por último, referira-se que o processo utilizado por Zoë é extremamente difícil de pôr em prática, não só por causa da concentração necessária para saber quando fazer o quê, ou seja para tocar um instrumento e gravar ou reproduzir as faixas nos tempos certos, tudo feito simultaneamente e em tempo real.

2.2 Visualização de Música em tempo real

Esta secção fala sobre aplicações para visualizar música em tempo real. Isto significa que a cada aplicação podemos dar como *input* qualquer música e a aplicação gerará uma visualização de acordo com o *input* dado, o que é óbvio em algumas aplicações em que, por exemplo, podemos escolher uma música e fazer o *upload* dela. Noutras situações, apenas verificando o código fonte é que podemos fazer esta afirmação e, apesar de não ser possível alterar a música na aplicação, isto poderia ser feito se alterássemos o código fonte.

Todas as aplicações descritas nesta secção utilizam a biblioteca *Three.js* para fazer os gráficos da visualização e as duas primeiras utilizam a *Web Audio API* para tratar de tudo o que é relativo a som, tanto reproduzi-lo como obter informação acerca da música, para que esta possa ser utilizada na parte gráfica.

2.2.1 *Loop Waveform Visualizer*

Loop Waveform Visualizer é uma aplicação que está disponível na Internet e que permite visualizar música na forma de anéis que partem desde o centro do ecrã e que se vão afastando progressivamente (Loop Waveform Visualizer). No *site* temos a opção de fazer o *upload* de uma música ou de usar uma pré-definida já existente na aplicação.

A visualização de música na forma de anéis ou *loops* não é algo novo, é aliás um conceito já bastante utilizado. Isto deve-se ao facto de associarmos facilmente o conceito de onda de som, que é bastante representada em duas dimensões (figura 5), a algo parecido com uma onda sinusoidal, a um anel, se pensarmos em três dimensões.

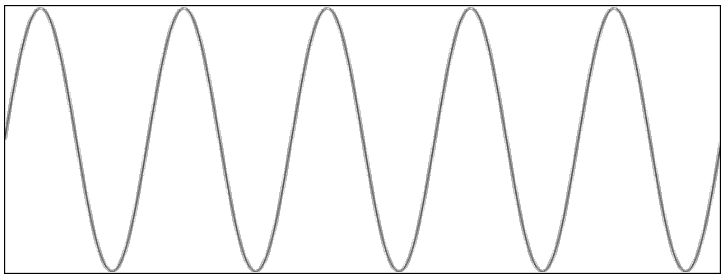


Figura 5 - Onda de som

Na aplicação não temos apenas os anéis a fluirem do centro para o exterior de forma linear. O *Loop Waveform Visualizer* recorre ao *RealtimeAnalyserNode* recentemente renomeado *AnalyserNode* (Web Audio API) na especificação mais recente da *WebAudioAPI*. Utilizando este nó é possível aceder a informação acerca da frequência e do *time-domain* em tempo real. Assim, e utilizando o nó referido, é usada a informação acerca dos níveis (quão alto é o som em termos de volume) para determinar o brilho, a espessura e o *Z displacement* dos anéis [(Loop Waveform Visualizer)]. Isto é bastante visível, basta-nos tomar atenção por exemplo ao ritmo da música que vem por defeito na aplicação e notar que, quando ele soa, a altura a que os anéis caem a partir do meio é maior (é fácil verificar isto uma vez que, na música que vem por defeito, o bombo encontra-se com o volume bastante alto em relação ao resto dos instrumentos). Passando agora para as restantes dimensões a serem manipuladas, temos por exemplo a cor. A cor dos anéis segue sempre a mesma ordem de transição, no entanto a velocidade a que isto se dá depende de algum parâmetro que não é de perceção imediata. Finalmente, vale a pena mencionar uma última especificidade desta visualização, que é a forma aleatória como os anéis ondulam (figura 6). Isto está também vinculado a uma qualquer variável ou conjunto de variáveis; no entanto, após uma rápida inspeção do código fonte, é fácil descobrir que aquela forma aleatória é conseguida através de uma função de *noise* e faz como que o resultado final seja mais variado e aleatório do ponto de vista visual.

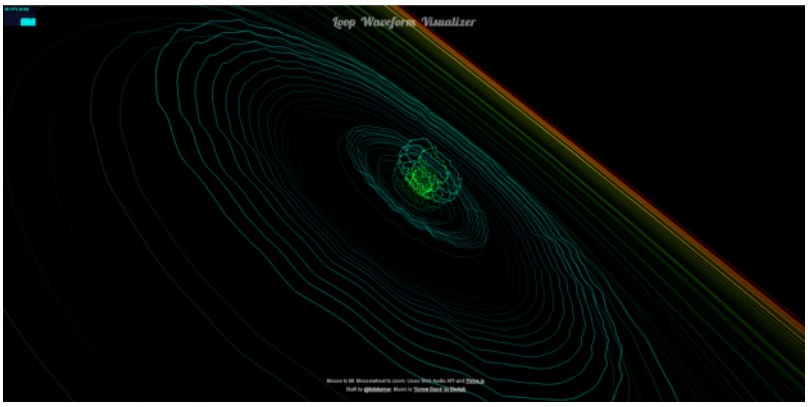


Figura 6 - Loop Waveform Visualizer *(Loop Waveform Visualizer)*

2.2.1 *Music Colour Particles*

Music Colour Particle é uma aplicação *web* criada com recurso a *WebGL*, nomeadamente *Three.js* (three.js), *Sparks.js* (spark.js) e *AudioKeys.js* (audio keys) que apresenta uma visualização baseada em partículas para a música *Arabesque* do compositor francês Claude Debussy (Music Colour Particles). O autor recorre a partículas que vão sendo lançadas a alturas diferentes pelo emissor de partículas consoante a música (figura 7). Esta altura é determinada pela informação acerca dos níveis das várias frequências que compõem o espectro. A cor das partículas muda ao longo do tempo e segue as cores do arco-íris. No entanto, o intervalo de tempo entre as cores mantém-se constante.

Seria interessante tentar não só fazer variar a altura a que as partículas são lançadas, como também fazer variar outros eixos segundo algum outro parâmetro ou ainda fazer com que as cores tivessem um papel mais importante e dinâmico, em vez de serem alternadas de forma constante.

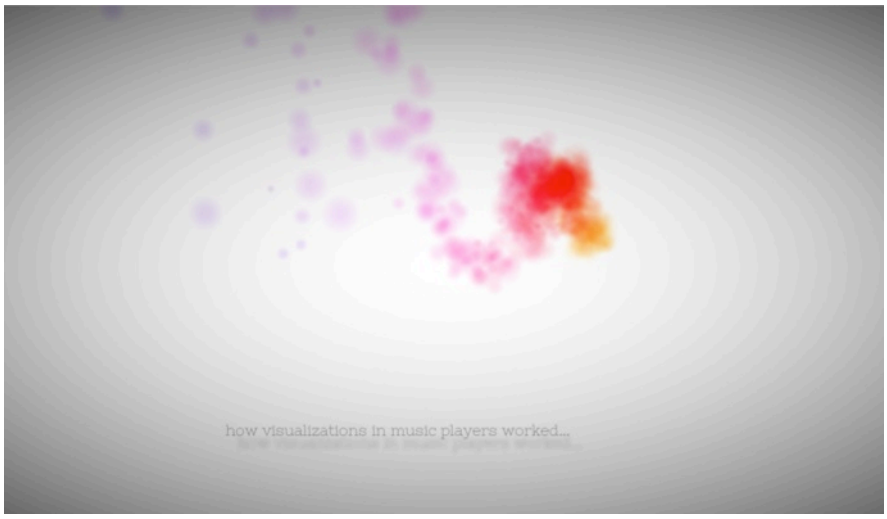


Figura 7 - Music Colour Particle no início da música (Music Colour Particles)

2.2.2 *Cube Visualizer*

Cube Visualizer é também uma aplicação que se encontra na *web* e que, dada uma música, produz dela uma visualização em tempo real (cubevis). Apesar de não ser possível, como era na aplicação anterior, fazer o *upload* de uma música directamente para a aplicação, e obter então a respectiva visualização, é possível alterar o código fonte para que tal aconteça, pois este foi feito de forma a receber como *input* uma música e não foi criado especificamente para a música que já se encontra na aplicação.

Esta visualização faz lembrar um equalizador de barras, com a diferença de ser feito em 3D e de seguir o aspecto de uma matriz, tal como indicado na figura 8.

Figure 8 - Equalizador de barras

Quando a aplicação inicia, a música começa imediatamente a tocar e os cubos que constituem a matriz começam a mover-se. Este movimento é sempre igual, consiste numa "onda" que vem das extremidades da matriz para o centro (figura 9).

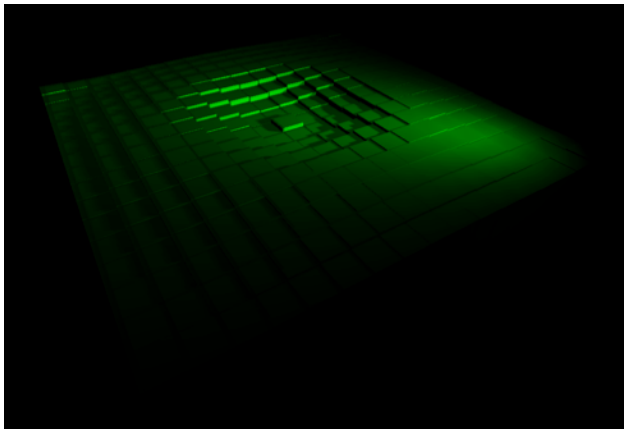


Figura 9 - Cube Visualizer [cubevis]

As dimensões que são manipuladas aqui são basicamente a velocidade a que os cubos sobem e descem para formar a onda, a velocidade a que mudam as cores da matriz e a velocidade a que a câmara circula.

Este é, no entanto, um exemplo de algo que poderia ter sido melhorado, pois o efeito conseguido pela visualização é um pouco monótono. Se tivesse sido seguida a abordagem por exemplo do equalizador de barras, talvez se conseguisse obter algo mais apelativo. A complexidade do código escrito e o facto de incluir um motor de física (que serve para movimentar os cubos durante a visualização) é completamente desaproveitado.

2.2.3 *A dive in Music*

A aplicação *A dive in Music* é sem dúvida a mais rica e complexa de todas as anteriormente referidas [A dive in Music]. Isto deve-se em parte ao facto de esta ser uma aplicação que contém mais de 20 visualizações, que vão sendo trocadas dinamicamente ao longo de uma ou várias músicas. Todas as visualizações presentes utilizam exclusivamente partículas (figura 10) e os parâmetros que regem toda a visualização podem ser mudados *on the fly*. A aplicação permite-nos não só fazermos o *upload* de músicas como usar, por exemplo, um microfone como *input* para a visualização, sendo que, por defeito, vem com uma *playlist* de várias músicas incluída.

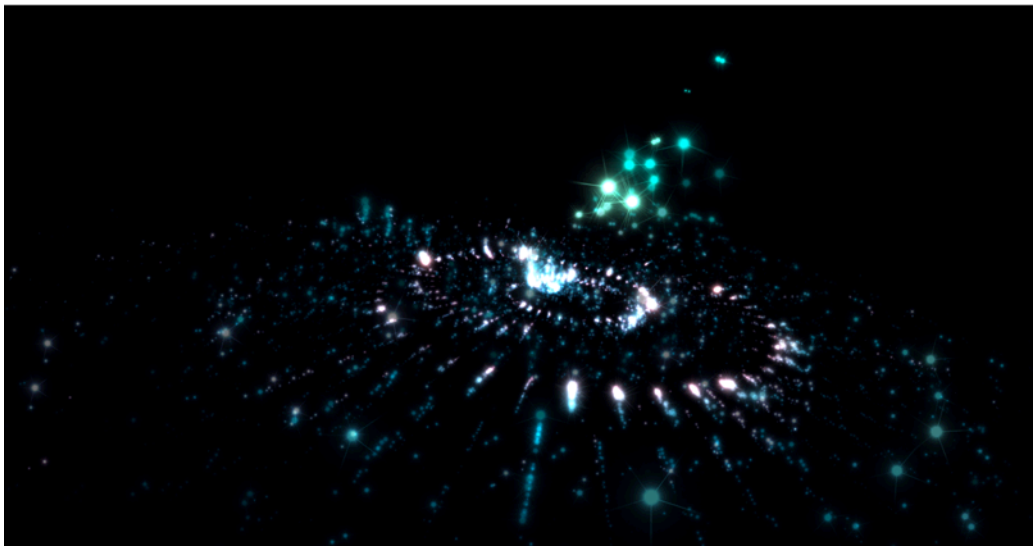


Figura 10 - Uma das visualizações presentes em A dive in Music [A dive in Music]

Todas as visualizações estão muito bem conseguidas, quer seja do ponto de vista técnico quer seja do ponto de vista gráfico. Quanto ao primeiro, o nível de *fps (frames per second)* é bastante bom para a quantidade de partículas que estão a ser processadas num dado instante, e, quanto ao segundo, as visualizações são bastante agradáveis à vista, bem como imersivas. A interface surpreende-nos um pouco ao início mas depois torna-se fácil navegar nela. Os parâmetros que são alterados na visualização são vários e dependem da visualização que está a ser reproduzida num dado momento. Como já foi referido anteriormente, a aplicação contém várias visualizações, por isso resta finalmente dizer que a forma como a visualização "acompanha a música" está muito bem conseguida, algo que não acontecia, por exemplo em outras aplicações analisadas anteriormente.

2.3 Visualização de Música Pré-processada

Nesta secção são apresentados exemplos de visualizações pré-processadas, ou seja, visualizações que não são geradas à medida que a música vai sendo reproduzida mas são visualizações em que os dados necessários para servirem de *input* são calculados previamente e apenas é apresentada a visualização ao utilizador. No caso dos vídeos descritos nesta secção, o facto de as imagens serem geradas *a priori* e só vermos o resultado final pode servir para que possam ser criadas visualizações mais exigentes em termos de computação gráfica.

2.3.1 *3D Waveform*

3D Waveform é uma aplicação web feita com recurso à biblioteca *Three.js*, já mencionada na secção anterior, que lida com os gráficos 3D [(3D Waveform)]. É uma aplicação que tem o conteúdo que vai servir de *input* pré-processado. Esse conteúdo dá-nos o tamanho dos paralelepípedos. O eixo ao longo do qual os paralelepípedos estão dispostos serve como eixo temporal e o tamanho que têm varia de acordo com algum padrão, aparentemente o nível de volume da música.

O interessante acerca desta visualização é que se assemelha muito a uma onda de som que é usualmente representada em 2D mas numa forma tridimensional, sendo que a forma escolhida pelo criador da aplicação foi o paralelepípedo em vez de uma forma circular como seria de esperar (figura 11).

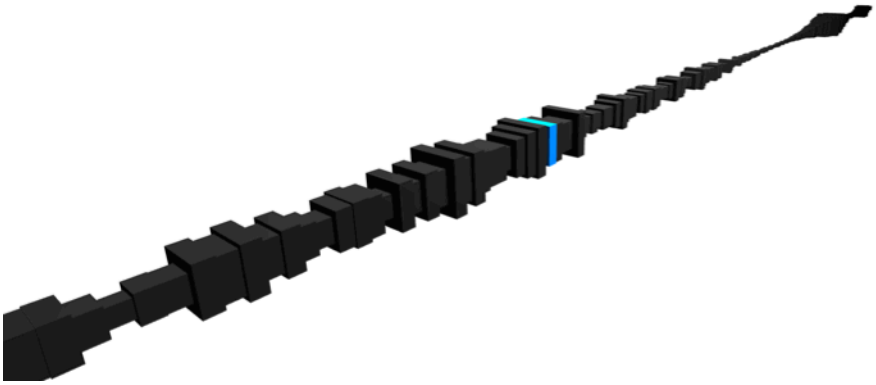


Figura 11 - 3D Waveform [(3D Waveform)]

2.3.2 *Ljósið*

Ljósið [(Ólafur Arnalds - Ljósið (Official Music Video))] é um video artístico, do *motion graphics artist* Esteban Diácono [(Esteban Diacono)], que foi feito como uma visualização para a música com o mesmo nome, do artista Ólafur Arnalds da editora *Erased Tapes Records* [(Erased Tapes Records)].

Este vídeo é bastante interessante não só do ponto de vista artístico como do ponto de vista técnico. Uma vez que não é feito em *real time*, e é por isso um vídeo e não uma aplicação, e também pelo facto de ter sido feito com programas de modelação 3D e de composição, pode obter-se um resultado muito mais complexo, rico e potencialmente com mais liberdade artística para quem o cria.

Para a elaboração deste vídeo foi usado em concreto o *software Particular* (Trapcode Particular 2.2), que é um *plugin* do conhecido programa de composição *visual effects* e *motion graphics After Effects* (After Effects). Este *plugin* é extremamente poderoso e é considerado um *standard* na indústria cinematográfica hoje em dia, tendo sido usado em vários filmes conhecidos. *Particular* foi especialmente construído como um *software* capaz de lidar com partículas de forma realista. O vídeo em análise é um ótimo exemplo disso, pois é inteiramente construído à base de partículas que se movimentam consoante a música (figura 12).

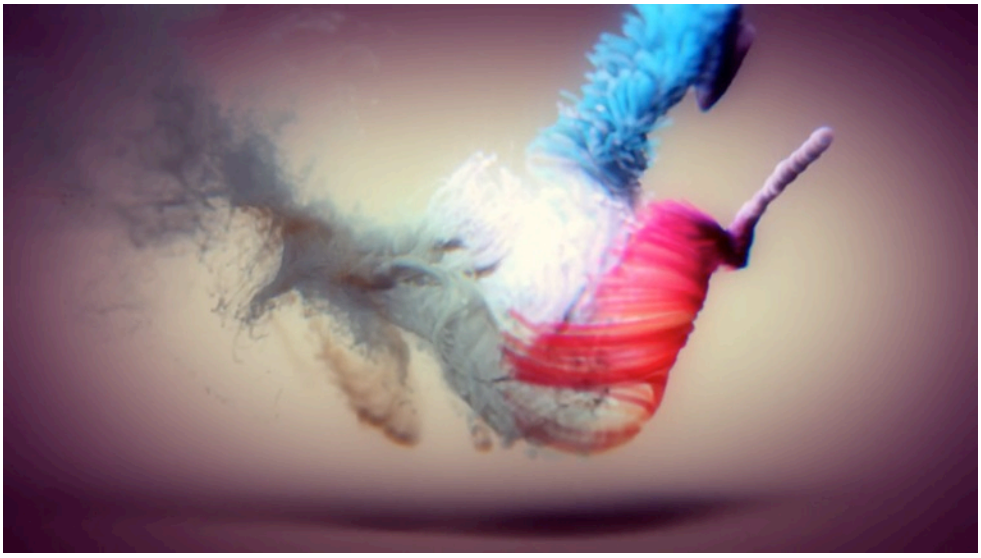


Figura 12 - Ljósið *de Esteban Diacono.* *(Esteban Diacono)*

Uma característica interessante da criação do vídeo é que, ao contrário de outros exemplos apresentados, neste em particular não sabemos se o artista que o criou usou algum tipo de informação acerca da música, como os níveis de volume ou informação acerca das várias frequências. O que podemos constatar facilmente é que no vídeo existem dois tipos de emissores de partículas a funcionar simultaneamente, sendo que o primeiro está ligado ao som do piano e o segundo ao som do violino. Existem outras partículas que não pertencem a nenhum dos dois emissores principais mas que têm uma funcionalidade meramente artística. O facto de os dois sons dos dois instrumentos terem, cada um, um tipo de visualização diferente, ainda que os dois interajam entre si, sugere que o autor do vídeo não se baseou meramente na informação que, por exemplo, um equalizador disponibilizaria. Ou existiu um tipo de análise mais complexa e aprofundada da música ou o autor não recorreu só a informação musical e ajustou manualmente as respetivas visualizações dos sons dos instrumentos. Visto que o vídeo tem uma finalidade meramente artística, esta premissa parece inteiramente razoável.

2.3.3 *Study on the procedural generation of visualization from musical input using generative art techniques*

A tese de mestrado *Study on the procedural generation of visualization from musical input using generative art techniques*, de Christopher Michael Garcia, explora a criação de visualizações com grande complexidade e com grande qualidade para música (GARCIA, 2011). Uma vez que estes são requisitos das visualizações, estas não foram feitas em tempo real. Assim, o processo de fazer a visualização divide-se em duas grandes partes: numa primeira é analisado um ficheiro *mp3* recorrendo a um programa feito em *Adobe Flash* (Flash) que, após ser parametrizado pelo utilizador, gera um simples ficheiro de texto onde se encontram valores correspondentes à análise feita, como por exemplo valores relativos a *beat detection*. A segunda parte do processo é feita no *software Maya* (Maya) da *Autodesk* (Autodesk) onde o ficheiro criado na primeira parte do processo vai servir de *input* para gerar uma visualização. Isto é feito com recurso a um *script Python* desenhado especificamente para esse propósito. Dentro do *Maya* temos acesso a uma interface própria que permite editar alguns parâmetros da visualização que vamos gerar.

Um dos objetivos do autor foi criar uma grande e óbvia correspondência entre a música que serve como *input* e a respetiva visualização, pois isto é algo que nem sempre acontece de forma clara nas aplicações anteriormente mencionadas. Para conseguir isto, o autor recorre à automatização que o seu programa fornece e que permite posteriormente o ajuste manual dos valores fornecidos pela análise. Deste modo, recai de uma certa forma sobre o utilizador que fará a visualização a responsabilidade de se assegurar de que esta tem a maior correspondência possível com a música original.

Seguindo o *pipeline* deste processo, o autor criou três tipos de visualizações diferentes, salvaguardando no entanto que estas são meramente exemplos, podendo ser criados muitos outros tipos de visualizações. Assim, a primeira visualização assemelha-se a um espectro analógico, animado numa espécie de pirâmide composta por vários cubos, tal como podemos verificar na figura 13.

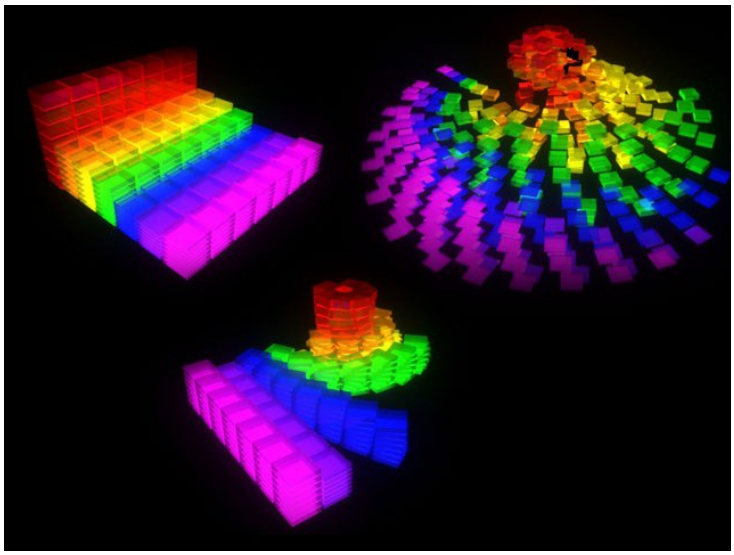


Figura 13 - Visualização nº1- Equalizador (GARCIA, 2011)

Para a segunda visualização foi utilizado o sistema de *fur* (pêlo) do *Maya*. Neste exemplo foram manipulados três elementos do pêlo: o seu comprimento, o *curl* (enrolar) e a sua *scraggle* (disseminação). Na figura 14 encontra-se um exemplo desta visualização.

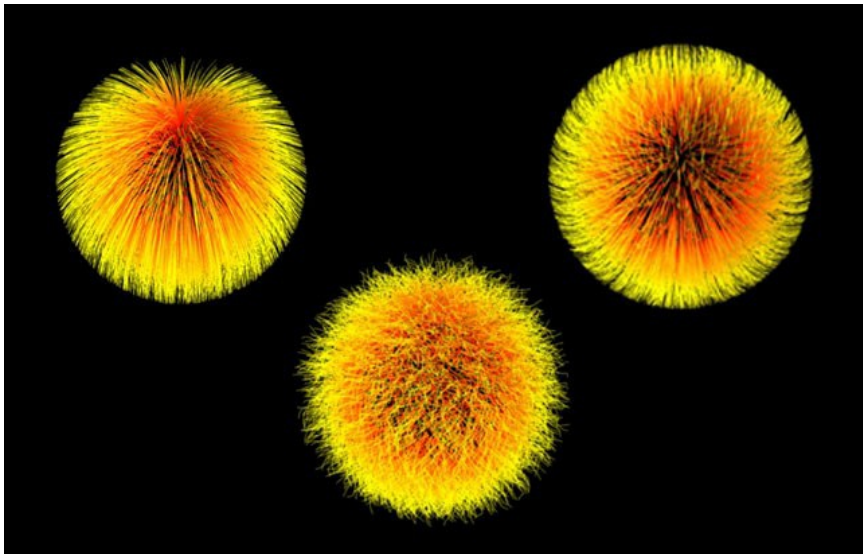


Figura 14 - Visualização nº2 – Fur (pêlo) (GARCIA, 2011)

Finalmente, para a terceira e última visualização foram usados os dados relativos à música para manipular o *riging* de uma personagem virtual (figura 15).

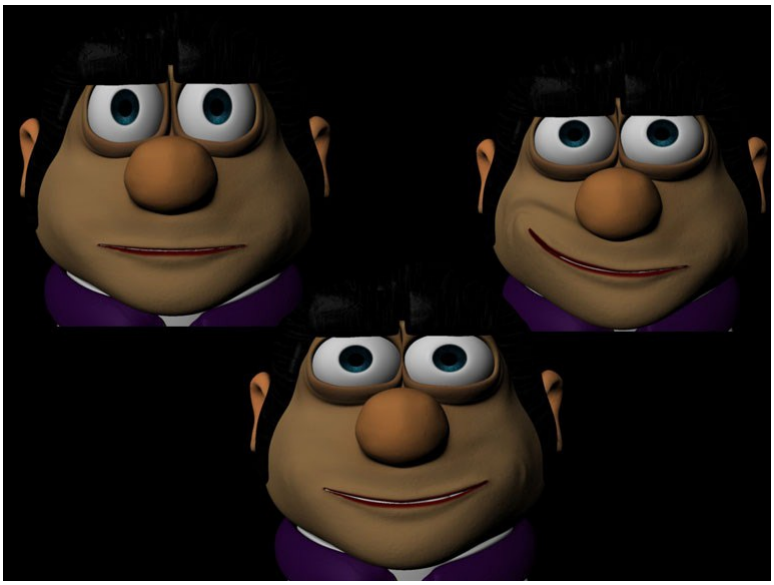


Figura 15 - Visualização nº3 – Riging (GARCIA, 2011)

2.4 Visualização e criação de Música Simultaneamente

Esta secção descreve várias aplicações onde é possível visualizar e criar música simultaneamente. As várias aplicações são feitas com recurso a várias tecnologias, algumas mais antigas, outras mais recentes, umas feitas em duas dimensões e outras feitas a três.

2.4.1 *ToneMatrix*

ToneMatrix (ToneMatrix) é uma aplicação musical *online* feita em *Abode Flash* e criada pela empresa *Audiotool* (Audio Tool). É uma aplicação que toma a forma de uma matriz bidimensional 16 x16 onde cada quadrado que vemos na figura corresponde a um som. Dependendo da posição do quadrado na matriz, o som é diferente, sendo que a matriz funciona da seguinte forma: o tempo é representado pelo eixo horizontal e o tom da nota encontra-se no eixo vertical. O utilizador pode carregar em um ou mais quadrados, ficando estes brancos sempre que se encontram selecionados (figura 16). Uma vez selecionados, os quadrados reproduzem um som que depende da sua posição na matriz.

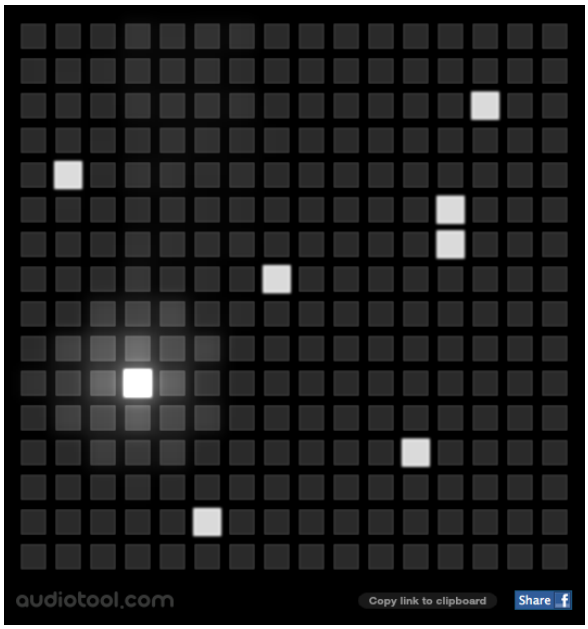


Figura 16 - ToneMatrix [(ToneMatrix)]

Os sons não são todos tocados ao mesmo tempo, existe uma ordem. Essa ordem vem do eixo horizontal, que representa o tempo, como foi já referido. Esse tempo não é contínuo, entre as notas que vão sendo tocadas existe um intervalo, ainda que pequeno, mas que faz com que, ao serem tocadas, as notas soem com uma cadência agradável. Relativamente ao eixo vertical, temos o tom que cada quadrado reproduz. Estes tons aparentemente não estão organizados numa escala cromática onde todos os doze semi tons existentes sejam tocados. Isto é notório devido à forma como os sons soam. Podemos assim assumir que os tons se encontram dispostos numa outra qualquer escala musical.

O facto de a aplicação forçar o utilizador a, em primeiro lugar, usar uma escala musical pré-definida e, em segundo lugar, não tocar os sons seguidamente mas sim com pequenos intervalos faz com que o resultado final soe como uma composição com a densidade musical de um acorde. Mesmo que alguém sem qualquer experiência carregue aleatoriamente nos quadrados da aplicação, vai obter um resultado bastante satisfatório em termos de som.

Resumindo, é de salientar que esta aplicação recorre a artifícios tão simples, explicados no parágrafo anterior, para fazer com que o utilizador se entusiasme pela aplicação e não se sinta frustrado com o facto de não conseguir produzir sons que soem agradavelmente. No fundo, o que aqui acontece é que a distância que existe entre

alguém que não tem qualquer conhecimento musical e a aplicação musical é reduzida usando pequenos artifícios.

2.4.2 *Beat Petite*

Beat Petite é uma aplicação *web* 2D feita inteiramente em *javascript* e que simula os vários sons que constituem uma bateria (BeatPetite) (figura 17). Esta aplicação foi feita por um Stuart Memo (Stuart Memo) que é um divulgador ativo da *Web Audio API* e tem inclusivamente várias aplicações disponíveis na sua página pessoal para qualquer pessoa experimentar.

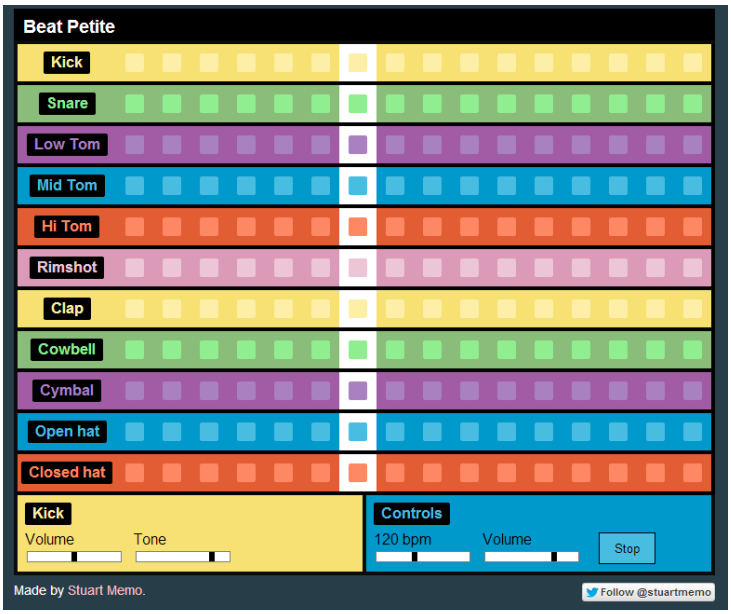


Figure 17 - Beat Petite (BeatPetite)

O objetivo desta aplicação é mostrar as potencialidades que a *Web Audio API* vem trazer à *web*. O conceito da aplicação é simples e é semelhante ao conceito empregue pela aplicação anterior *ToneMatrix*. Assim, temos dois eixos, o eixo horizontal que é o eixo do tempo e o eixo vertical que é o eixo que contém os vários tipos de sons. De notar que, em relação à aplicação *ToneMatrix* analisada anteriormente, aqui temos sons diferentes. Já na aplicação anterior era apenas o tom do som que mudava.

Para cada tipo de som existem 16 quadrados que podem ser ativados, sendo que, quando um quadrado está ativado, emite um som assim que for a sua vez. Existe uma barra branca que segue da esquerda para a direita em *loop* e que representa o tempo.

Este tempo tem a forma de *bpm* (*beats per minute*) e pode ser alterado fazendo com que a barra ande mais rápido ou mais devagar. A barra branca vertical vai passando por todos os quadrados dos vários sons. Se o quadrado está selecionado, esse som é reproduzido. Existem mais parâmetros que se podem alterar, como os volumes dos sons individuais ou o tom em que eles se encontram.

Esta é uma forma tradicional de representar em 2D uma aplicação de composição musical com os dois eixos, um representando o tempo e outro as *tracks*. Este é um exemplo bastante simples de uma aplicação musical mas, como já foi referido, o propósito é mostrar as potencialidades da *Web Audio API,* como, por exemplo, a baixa latência e alta precisão que permitem que os sons sejam tocados num instante exato.

2.4.3 *ToneCraft by* DinahMoe

ToneCraft (Tone Craft) é uma aplicação musical tridimensional disponível na *web* e feita com recurso à biblioteca *Three.js* e *Web Audio API*. Foi feita por Dinah Moe e, segundo ele, foi inspirada na aplicação *ToneMatrix* já mencionada anteriormente. Esta aplicação apresenta-se um pouco como a mistura das duas aplicações anteriormente analisadas; por um lado temos a parte matricial e dos vários tons de um mesmo som de *ToneMatrix*, por outro, temos a parte dos vários sons vindos de diferentes instrumentos da aplicação *Beat Petite*. Esta aplicação é especialmente interessante porque já não toma uma abordagem 2D, mas sim 3D, o que permite explorar mais graus de liberdade (figura 18).

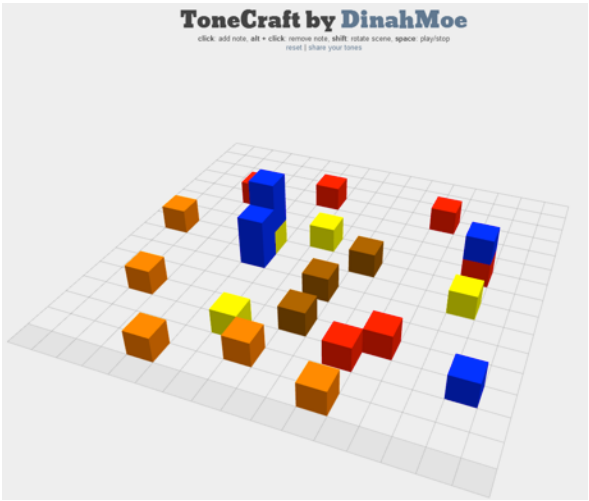


Figura 18 - ToneCraft (Tone Craft)

A aplicação é bastante parecida com as restantes analisadas, só que temos cubos em vez de quadrados devido à terceira dimensão. Tem por base uma matriz 16x16 a que se junta mais uma dimensão que permite empilhar cubos em cima de cubos. As cores dos vários cubos representam vários sons e existem sete cores diferentes que podemos escolher.

Assim, temos o eixo do X a corresponder ao tempo, o eixo do Y a corresponder ao *pitch* (ou tom) do som e temos o eixo do Z que nos permite colocar vários cubos com o mesmo som ou com sons diferentes em cima uns dos outros.

Finalmente, resta dizer que esta aplicação possui uma grande liberdade em termos artísticos, tal como acontece com a aplicação *BeatPetit*. São usados os pequenos artifícios já referidos, os quais, com a possibilidade que a terceira dimensão traz, melhoram e enriquecem a experiência de utilização.

2.4.4 *SOUND BOUND: Making a Graphic Equalizer More Interactive and Fun*

Sound Bound (Seungki Kim, 2013) consiste num equalizador gráfico interativo que permite a manipulação de música à medida que esta vai sendo reproduzida. Neste aspeto é menos uma aplicação que permite criar música a partir do zero do que uma que permite manipulá-la. O principal objetivo da aplicação, segundo os autores, é melhorar a experiência do utilizador comum ao ouvir música.

Assim, uma típica interação com a aplicação consiste em reproduzir uma música e observar a respetiva representação num equalizador gráfico. Depois o utilizador pode, através de uma interface *touch,* criar bolas num espaço vazio da interface. Quando as bolas caem e tocam nas barras do equalizador (com recurso a um motor de física que torna isto possível), o som correspondente à barra ou barras em que a bola tocou é amplificado. Quanto mais tempo o utilizador estiver a tocar no ecrã maior a bola fica e consequentemente maior é a distorção que se vai dar (figura 19).

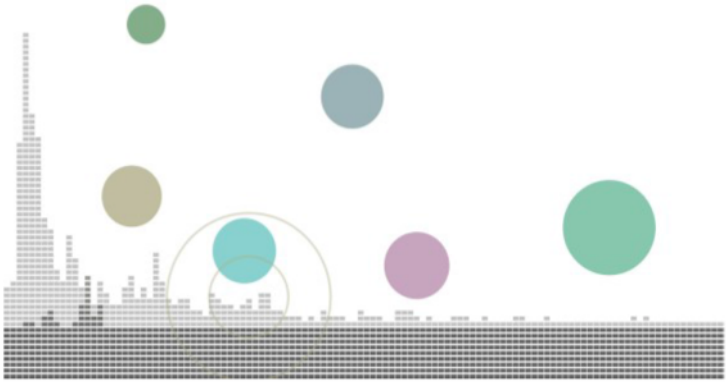


Figura 19 - Sound Bound (Seungki Kim, 2013)

O outro modo de interação da aplicação consiste em usar sons de instrumentos em vez da amplificação. Os sons dos instrumentos podem vir de sons previamente gravados ou por via de um microfone ligado ao computador.

Esta aplicação de cariz lúdico pode ser considerada um jogo, em vez de algo de cariz mais profissional que pretenda ter um propósito de composição musical.

2.5 Sumário

Foram apresentados neste capítulo vários artistas musicais e aplicações interativas. Quanto aos artistas, foram apresentados vários que pertencem a estilos musicais distintos mas que têm um processo de composição comum. Relativamente às aplicações interativas, foram apresentadas várias, umas com gráficos em três dimensões outras com duas dimensões, que são usadas tanto para apresentar uma visualização como para compor e tocar música interativa e visualmente.

No próximo capítulo será apresentado o trabalho desenvolvido para criar uma aplicação que tem o mesmo processo base dos vários artistas que foram discutidos anteriormente, e que oferece uma visualização tridimensional que reage à música que está a ser reproduzida. Essa visualização foi criada tendo em conta os exemplos expostos acima como ponto de partida e tentou dar o seu próprio contributo inovador.

Capítulo 3

Trabalho desenvolvido

Neste capítulo descreve-se o trabalho desenvolvido que integra quatro componentes principais:

a) áudio, que trata da reprodução e gravação dos vários sons, da combinação dos sons gravados e da marcação do tempo da forma mais exata possível;

b) interação para a reprodução de sons;

c) interface para construção, composição de música e marcação de tempo;

d) visualização tridimensional da música criada.

A aplicação resultante poderá ficar disponível numa página web, que, uma vez *online*, permitirá que qualquer pessoa possa aceder à aplicação de forma rápida e fácil, não sendo preciso fazer qualquer *download* adicional. Esta foi uma das decisões que mais pesou na escolha das tecnologias a usar para desenvolver este projeto. Deste modo, é apenas preciso possuir o dispositivo Leap Motion e aceder a um endereço *web* para usufruir da aplicação.

A aplicação foi desenvolvida na linguagem *javascript* para o browser *Google Chrome* em ambiente *Windows,* que são neste momento, o único browser e sistema operativo que suportam todas as funcionalidades de *HTML5* necessárias para a aplicação funcionar. As funcionalidades *HTML5* utilizadas na aplicação foram maioritariamente duas: *WebGL* (Mozilla), que dá a possibilidade de mostrar e manipular gráficos 3D no *browser* (para isto foi utilizada a biblioteca *Three.js* (Three.js) que abstrai muitos detalhes de implementação de *WebGL* e que foi referida no capítulo anterior) e a *Web Audio API* que permite processar e sintetizar áudio, também já mencionada.

3.1 Áudio

A componente de áudio foi a que se demonstrou mais desafiante. Nesta componente podemos encontrar duas vertentes. Por um lado desenvolver um metrónomo para a aplicação ser capaz de marcar o tempo de forma exata. Por outro lado, disponibilizar vários sons, organizar as várias *tracks* que compõem a aplicação e lidar com o processo de reprodução e gravação de *tracks*.

Tendo em conta a tecnologia utilizada, foi difícil conseguir acomodar estes requisitos e ao mesmo tempo ter em conta aspetos de desempenho, isto é evitar que os atrasos no som sejam perceptíveis aos utilizadores.

3.1.1 Grafo de áudio

Toda a componente áudio foi feita com recurso à *Web Audio API*, e o paradigma principal usado por esta API para lidar com o áudio é um grafo composto por vários tipos diferentes de nós que estão interligados entre si e que fazem o encaminhamento do áudio desde a(s) fonte(s) de som até à saída. Concretamente, nesta aplicação foram usados diversos tipos de nós interligados de várias formas para poder suportar todas as operações que a aplicação disponibiliza.

Todo o grafo é construído no momento de inicialização da aplicação, isto significa que todos os nós são criados e que são estabelecidas todas as relações entre eles. Passada esta fase de inicialização são manipuladas as várias propriedades que são disponibilizadas por cada nó, um exemplo disto seria mudar o volume de uma *track*.

Na imagem 20 encontra-se o grafo usado na aplicação.

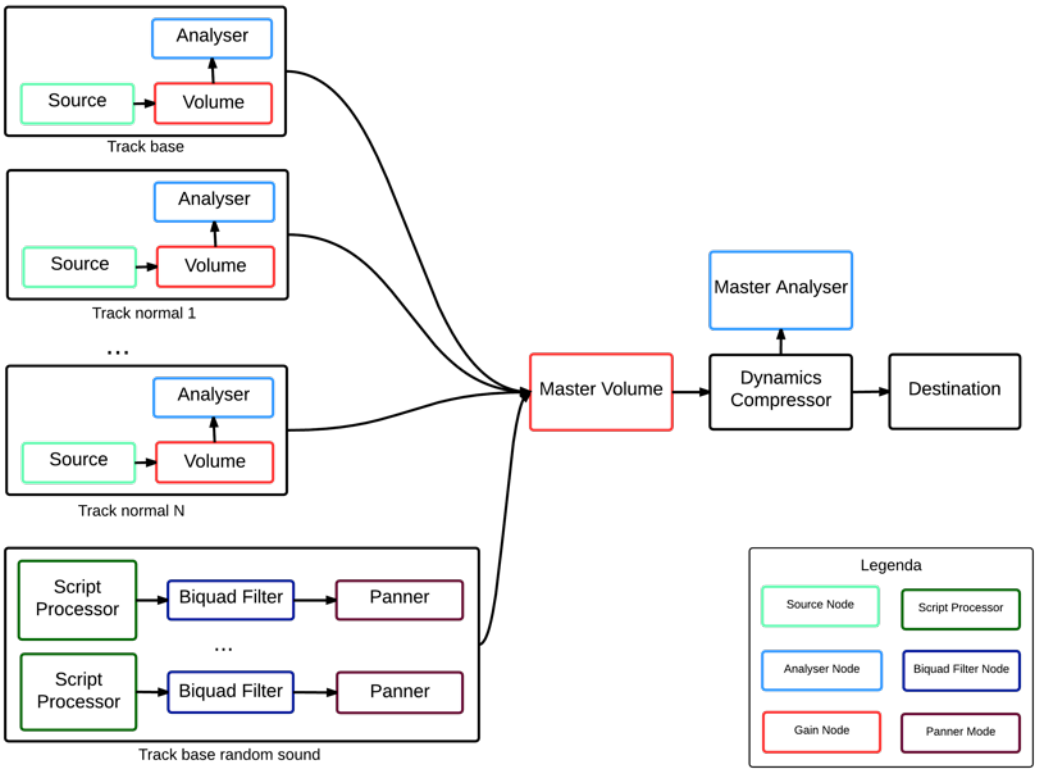


Figura 20 - Grafo de encaminhamento de áudio da aplicação

Podemos dividir o grafo em duas grandes partes, os nós que são comuns a todas as *tracks* e que na imagem 20 se encontram à direita (os quatro nós: *Master Volume* (Web Audio API - Gain Node), *Master Analyser* (Web Audio API - Analyser Node), *Dynamics Compressor* (Web Audio API - Dynamics Compressor) e *Destination* (Web Audio API - Destination Node)) e as *tracks* em si mesmas (secção da esquerda). Relativamente às *tracks* podemos considerar dois tipos, as *tracks* base e as normais (serão abordadas em maior detalhe mais a frente). Dentro das *tracks* base temos *tracks* que utilizam dois tipos de sons, uma delas que usa sons reais à semelhança das *tracks* reais e outra usa sons sintetizados.

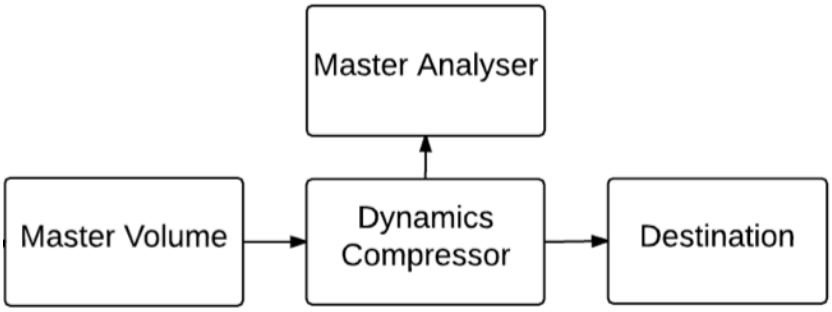


Figura 21 - Detalhe do grafo de fluxo de som. Secção comum a todas as tracks

Assim, relativamente ao tronco comum, existem quatro tipos de nós: um *Destination Node*, um *Dynamics* um *Compressor Node*, um *Analyser Node* e um *Gain Node* (figura 21). O *Destination Node* é o nó à direta e é o nó a que todos os outros nós se ligam direta ou indiretamente. É o nó que representa a saída de áudio do computador (ou seja o som que o utilizador de facto ouve) e que em ultimo caso pode ser os altifalantes do computador ou um conjunto de *headphones*. Relativamente ao *Dynamics Compressor,* este tem como função prevenir a distorção do som e para isso aumenta os sons que são mais baixos e diminui os sons mais altos. Este nó é o único que se liga diretamente ao *Destination*. A seguir temos um *Analyser Node*, designado na aplicação como *Master Analyser*, que analisa o som que passa por ele e disponibiliza essa informação ao programador para que este possa usá-la. O. A ligação é feita com o *Dynamics Compressor* a ligar ao *Master Analyser* para que a análise feita ao som seja o som final que o utilizador final está de facto a ouvir. Se a ligação fosse feita com outro nó que não este isso não se verificaria. Finalmente existe um *Gain Node* que é designado como *Master Volume* pois é este nó que controla o volume geral da aplicação. É também a este nó que está associado o *slider* da interface que controla o volume (a interface será abordada em maior detalhe num capítulo posterior).

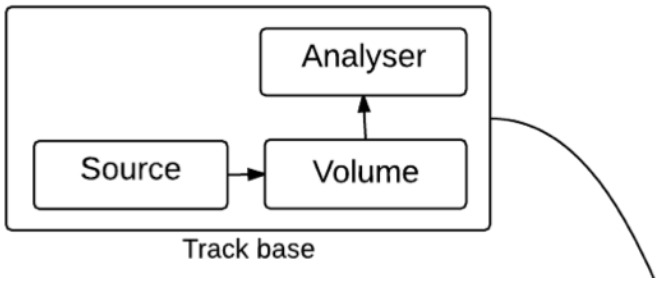


Figura 22 – Secção do grafo relativo à base track

Passando agora para as *tracks* base (figura 22), existem na aplicação duas *tracks* base, uma tem o som de um bombo de uma bateria e reproduz um som real e uma outra reproduz um som contínuo e envolvente mas sintetizado. A *track* base, que utiliza o som do bombo, é composta por um *Gain Node*, designado Volume, e um *Analyser Node,* nomeado simplesmente *Analyser*. O Volume serve de *input* ao *Analyser* e está ligado ao *Master Volume* referido anteriormente e pertencente à secção comum a todas as tracks. O Volume, tal como o nome indica serve para controlar o volume desta track específica e o *Analyser* serve para fornecer informações acerca do som que está a passar por esta track. São estas informações que são usadas posteriormente para efetuar a animação da visualização que será abordada posteriormente. Para além destes nós, falta apenas referir o *Source Node* (Web Audio API - Source Node)(*Source*) que contém o som que é de facto reproduzido. Este *Source* liga com o *Volume* tal como é apresentado na figura 22.

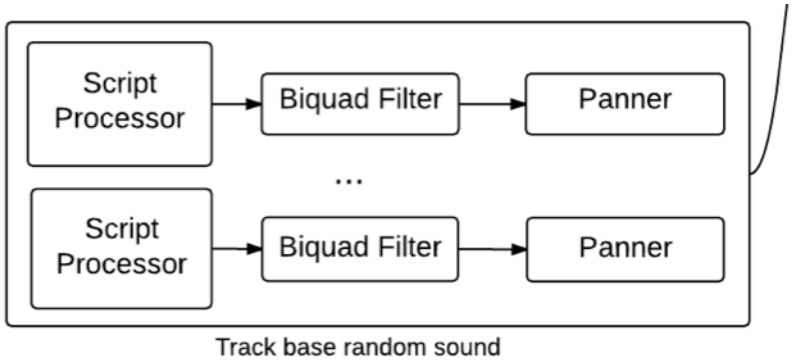


Figura 23 - Conjunto dos nós de geradores aleatórios de som da track base

A restante *track* base, que contém o som contínuo, é mais complexa. Não foi criada de raiz, mas sim adaptada de uma biblioteca existente. Ela produz som ao utilizar até 20 geradores aleatórios de som. Cada um destes geradores de som aleatório utiliza,

em primeiro lugar, um nó *Sctipt Processor* (Web Audio API - Script Processor Node) para gerar sons de maneira aleatória. De seguida, passa o som aleatório gerado por um *Biquad Filter Node* (Web Audio API - Biquad Filter Node), que está configurado para o modo BANDPASS, e que limita o intervalo de frequências que passam pelo filtro. Finalmente é usado um *Panner Node* (Web Audio API - Panner Node)que faz com que o som proveniente do filtro mude de direção. Este *Panner Node* é essencial, pois a direção do som é mudada aleatoriamente a cada meio segundo, o que faz com que, quando se ouve o som este pareça estar sempre a mudar de direção e faz ainda com que o som soe aleatório. É assim que cada um dos 20 geradores aleatórios de som funciona, utilizando estes três nós desta maneira específica para produzir som.

Cada um dos geradores de som aleatório possui uma frequência que serve de parâmetro para o *Biquad Filter* e cada gerador é criado com uma frequência diferente, que é selecionada de forma aleatória de um conjunto restrito de frequências. Isto faz com que o som final obtido seja agradável e que estes não destoem uns dos outros. Não nos esqueçamos do seguinte: a aleatoriedade da direção do som produzida pelo *Panner Node* é individual de cada gerador aleatório de som, ou seja, cada um possui uma aleatoriedade própria e, cada gerador aleatório de som possui também cada um, uma frequência, e consequentemente, uma nota própria. É a junção de todos estes factores que ultimamente faz com que este som soe bastante cheio e envolvente.

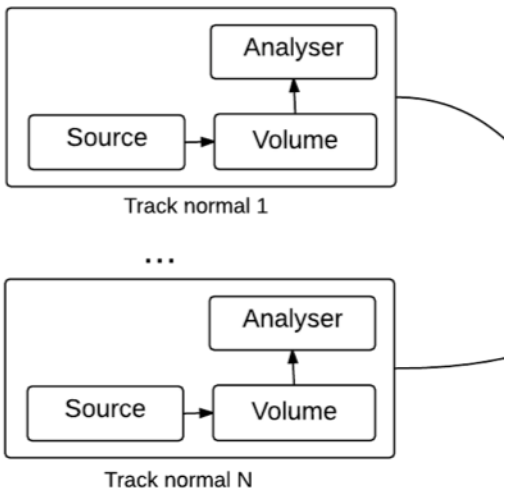


Figura 24 - Organização das tracks normais

Finalmente temos as *tracks* normais, estas *tracks* são compostas por três nós, da mesma forma que a *track* base inicial possuía, um *Source Node*, um *Gain Node* e um *Analyser Node*, sendo que o primeiro liga ao segundo e o segundo ao terceiro e ao *Master Volume* (figura 24). Existem presentemente apenas cinco *tracks* normais, mas poderão ser facilmente adicionadas mais numa versão futura da aplicação.

Falta ainda referir uma particularidade comum a todos os *Source Nodes*. Até agora tratou-se cada um deles como se fosse único e imutável ao longo do ciclo de vida da aplicação mas isto foi feito apenas para efeitos de simplificação. Ao contrário do que seria de esperar e por motivos de desempenho e arquitectura da *Web Audio API*, cada *Source Node* pode apenas ser reproduzido uma vez. Após a reprodução, e caso se queira tocar de novo o som (operação que é feita muitas vezes nesta aplicação), é necessário criar um novo *Source Node* que contenha o som que tem que reproduzir e por fim reproduzi-lo. Isto, no entanto, não significa que o som tenha de ser outra vez carregado para memória. Essa operação é feita apenas uma vez com o inicializar da aplicação.

3.1.2 Metrónomo, marcação de tempo e desempenho

Ao começar o módulo de áudio, foi claro que a marcação do tempo e a necessidade de construir um metrónomo que fosse capaz de o contar de forma precisa e igualmente espaçada seria crucial. Assim, e nunca esquecendo que toda a aplicação é em tempo-real, pensou-se desde o início nas implicações em termos de desempenho. Desde logo, pelo facto de a aplicação correr num *browser* fica limitada em termos de precisão do seu relógio. Apesar de o *Google Chrome*, *browser* para o qual a aplicação foi desenvolvida, ter o valor de atraso do relógio mais baixo quando comparado aos restantes, ainda assim esse atraso é significativo quando se trata de som. Por isso, foram usadas várias técnicas para tentar mitigar os seus efeitos.

A forma como o metrónomo funciona na aplicação é a seguinte: com base no BPM (*beats per minute*) e no número de notas que é possível dar num dado compasso, é definido um número de milissegundos de intervalo entre cada "batida" da música; cada vez que passa esse intervalo de tempo, o metrónomo "acorda" e sabe que necessita de efetuar uma série de ações; essas ações em *javascript* traduzem-se na execução de uma função *callback*. Esta função é responsável por efetuar uma série de ações que constituem uma secção crítica da aplicação, ou seja, ações que têm de ser feitas num

certo espaço de tempo e com o menor número possível de atrasos. Assim, a função *callback* é responsável por:

a) começar a gravar uma *track*, caso tenha sido efetuada essa ação pelo utilizador;
b) terminar de gravar uma *track*;
c) actualizar o contador que é responsável por saber quando parar de gravar uma *track*;
d) reproduzir som se o utilizador quer tocar o som de determinada *track*;
e) reproduzir os sons das *tracks* base;
f) caso haja alguma *track* com algum som já gravado, e se for o momento de reproduzir esse som, de facto reproduzi-lo;
g) por último, acertar os valores da contagem das notas que servem para marcar o tempo (estes valores são apresentados na interface gráfica e serão explicados numa secção posterior).

Como já foi referido anteriormente, existe um atraso de alguns milissegundos no relógio das aplicações que usam o *browser*. Isto faz com que o intervalo de tempo que passa entre cada "acordar" do metrónomo não seja igual. O efeito cumulativo de todos os atrasos causa, naturalmente, problemas acrescidos em termos de desempenho.

Por esta razão, o que se fez para resolver este problema foi ajustar o número de milissegundos que leva o próximo "acordar" do metrónomo, não para o seu valor esperado, mas sim para um valor que tenha em conta o atraso anterior. Por exemplo, supondo que o metrónomo deveria "acordar" no instante 100 mas que, devido ao atraso, apenas acorda no instante 104, e supondo ainda que o intervalo entre cada "acordar" do metrónomo é de 100ms, então o próximo instante seria provavelmente o 208 (104 do instante original mais os 100ms de intervalo mais o atraso que este segundo "acordar" teria, 104 + 100 + 4 = 208). Ora, este não é o efeito pretendido. O esperado seria o próximo "acordar" do metrónomo aos 200ms ou, na pior das hipóteses, no instante 204, que tem apenas o erro deste instante e não o erro acumulado do "acordar" inicial.

Para conseguir os tempos o mais regulares possível foram usadas algumas técnicas. A primeira foi ter em conta o atraso do *callback* passado e chamar o próximo com uma correção. No exemplo anterior, isto queria dizer que o segundo *callback* seria chamado não com 100ms de atraso mas sim com 96ms (100ms menos 4ms de atraso do

callback anterior). Assim elimina-se o erro cumulativo e fica-se apenas com o erro de cada *callback*, sendo este ajustado para o próximo *callback* que é feito.

A técnica descrita acima reduz para níveis aceitáveis o erro de relógio entre as chamadas de *callback*. Mas dentro da função *callback* que é sistematicamente chamada é preciso também ter cuidados. Nesta função são feitas operações como começar a gravar, parar de gravar, tocar o som de várias *tracks*. Se estas operações não forem feitas em sequência resultam em discrepâncias no áudio posteriormente ouvido pelo utilizador. Assim, existe uma zona crítica na qual as operações têm de ser feitas consecutivamente. Para isso, é necessário que as operações críticas sejam previamente preparadas.

Testou-se a utilização de *threads* (Mozilla, Using web workers) para esta parte do áudio, mas verificou-se a sua ineficiência, pois a comunicação entre a *thread* principal e a secundária implicaria demasiado atraso, e isto seria inaceitável numa aplicação que corre em tempo real e que lida com áudio.

3.1.3 Tracks

A aplicação é constituída por várias *tracks*, tocando cada uma um som específico. Esse som é obtido através de um ficheiro *.wav* carregado quando a aplicação inicia e que contém o som de um instrumento ou uma parte de um instrumento (maioritariamente sons de bateria: bombo, tarola, pratos). Existem tipos diferentes de *tracks*: as *tracks* base e as *tracks* normais.

A diferença entre as duas é o nível de controlo que o utilizador tem sobre elas: nas *tracks* base o utilizador não tem qualquer controlo, elas são tocadas automaticamente desde que a aplicação é iniciada até que é encerrada. O objetivo destas *tracks* é dar ao utilizador uma base sobre a qual trabalhar. Presentemente existem duas *tracks* base, uma que marca o ritmo com o som de um bombo de uma bateria e uma que dá um som envolvente e contínuo. Passando para as *tracks* normais, estas podem ser controladas pelo utilizador, concretamente através do dispositivo *Leap Motion*. Será explicado em mais detalhe na secção seguinte como é que é feita esta interação.

Os sons para cada uma das *tracks* foram escolhidos tendo em conta como soam isoladamente e em conjunto e – mais importante ainda – pela sua duração. Tal como já foi explicado anteriormente, existe um intervalo de tempo específico que separa as

chamadas consecutivas de um *callback* e os sons, ao serem escolhidos, não podem exceder esse limite para que não se sobreponham uns aos outros.

3.1.4 Gravar som

Para gravar o som foram tentadas várias abordagens. A primeira foi gravar o som usando uma biblioteca que utilizava nós *javascript* e *threads*. O resultado não obtinha a precisão necessária para começar a gravar nem para parar num momento exato, devido à latência inerente às *threads* em *javascript* e aos nós *javascript* que são usados na *Web Audio API*. Foram também usadas várias variações desta abordagem, quer usando apenas os nós *javascript*, sem as *threads*, quer usando apenas *threads* sem os nós *javascript* presentes. Nenhuma das tentativas teve resultados que tivessem um nível de desempenho aceitável.

Depois de várias iterações de tentativa e erro, chegou-se finalmente a uma solução aceitável em termos de desempenho: não gravar o som que saía de determinada *track* mas sim gravar o momento em que o som era tocado. Isto significa que, na prática, o que acontece é: quando a aplicação está em modo de gravação, sabe que está a gravar determinada *track* e sabe quando é tocado um som; assim, é gravado num *array* um valor booleano. Se o valor for verdadeiro, quando for altura de reproduzir o som posteriormente, este é de facto reproduzido; se o valor for falso, o som não é reproduzido.

Só pode ser gravada uma *track* (e consequentemente um som) de cada vez. O objetivo é imitar o processo de gravação/reprodução dos artistas descritos no capítulo anterior.

3.2 Interação para a reprodução de sons

A interação entre o utilizador e a aplicação no que respeita a reprodução dos vários sons que se encontram nas várias *tracks* é feita através do dispositivo Leap Motion (figura 24).

Figure 25 - O dispositivo Leap Motion

Ao utilizar-se este dispositivo para fazer tocar sons e para controlar a parte musical da aplicação, foi necessário ter em conta novos aspetos de interação que são radicalmente diferentes de, por exemplo, um rato e teclado.

Na aplicação o utilizador consegue controlar sons de ritmo, ou seja, sons que não se prolongam no tempo, por exemplo, o som de um tambor de uma bateria. Um contraexemplo disto seria o som de um violino, onde o músico pode prolongar o som de uma nota por algum tempo. Assim, visto que estes sons terminam rapidamente, escolheu-se dar ao utilizador a possibilidade de controlar dois parâmetros relativamente a eles: primeiro, o número de vezes que é tocado (tendo em conta os BPMs da aplicação), considerando sempre que o momento tocado é controlado pelo metrónomo da aplicação para que os sons "encaixem" todos no sítio correto; em segundo lugar, o volume do som.

Uma vez selecionada uma *track* e consequentemente um som, o utilizador pode tocar esse som. A mão esquerda controla o número de vezes que o som é tocado. Opcionalmente, a mão direita pode controlar o volume do som que está a ser tocado. Para conseguir identificar as várias posições das mãos foram definidos limites, tanto horizontais como verticais. Considerou-se um referencial de mão direita como apresentado na figura 26.

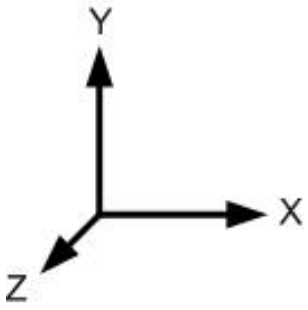


Figura 26 - Referencial tridimensional

Assim, definiu-se um plano vertical, paralelo aos eixos dos *yy* e dos *zz* colocado ao meio da imagem e que separa as zonas de ação de cada uma das mãos (figura 27).

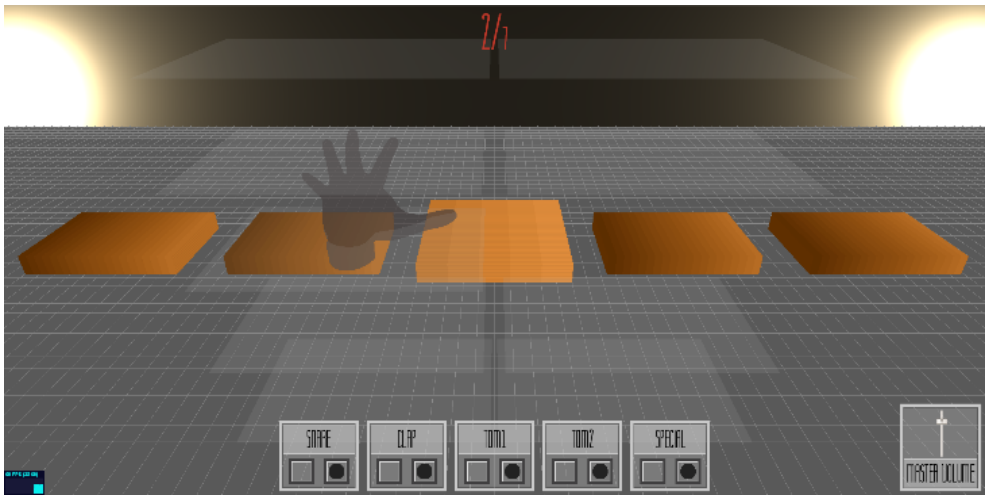


Figura 27 – Imagem da aplicação sem nenhuma track *selecionada*

Para a mão esquerda foram definidas verticalmente quatro zonas, correspondendo a cada uma ações distintas:

- a mais alta impede que seja tocado qualquer som (z1);
- a seguinte toca o som a uma velocidade normal, e, de acordo com a batida do metrónomo (z2);
- abaixo desta, o som é tocado a uma velocidade duas vezes superior ao normal (z3);
- e finalmente, na última zona, o som é tocado a uma velocidade quatro vezes superior à normal (z4).

A velocidade a que o som é tocado está diretamente relacionada com os BPMs do metrónomo e do número de notas que é possível tocar, bem como com os momentos em que podem ser tocadas.

Na tabela 1 assinalam-se com "x" as células correspondentes aos momentos em que os sons podem ser reproduzidos consoante a zona (z1, z2, z3 ou z4) em que a mão esquerda do utilizador se encontra.

z1																
z2	x				x				x				x			
z3	x		x		x		x		x		x		x		x	
z4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Zona vs tempo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Tabela 1 - Momentos em que os sons podem ser reproduzidos em cada zona de interação.

A mão direita controla o volume do som tocado. Para esta mão foram igualmente definidos limites que representam diferentes ações. Assim, as zonas verticais utilizadas para a mão direita consistem no seguinte: a zona mais alta corresponde ao volume mais alto (80%), a zona mais baixa corresponde ao volume mais baixo (10%) e na zona intermédia é feita uma proporção entre as coordenadas da mão do utilizador e o volume. De notar que as zonas nos extremos, mais alta e mais baixa, ocupam cada uma 25% do espaço e a zona intermédia ocupa os restantes 50% do espaço.

Tal como foi mencionado anteriormente, o utilizador pode escolher utilizar apenas a mão esquerda (a que controla se o som é tocado ou não) e, se assim for, o volume do som corresponde ao último volume utilizado pela mão direita ou, caso a mão direita nunca tenha sido utilizada naquela *track*, é usado um volume por omissão.

Uma vantagem que advém disto é a possibilidade de o utilizador escolher o volume desejado para tocar determinado som com a mão direita e depois usar unicamente a esquerda para escolher os momentos exatos em que quer que o som seja tocado.

Na figura 27 é possível ver a mão virtual de um utilizador que está a usar a aplicação. São visíveis também os limites discutidos anteriormente. Cinco planos dividem o lado esquerdo de modo a criar quatro zonas e quatro planos dividem o lado direito para criar três zonas. De notar, no entanto, que na imagem nenhuma *track* está selecionada, e por isso tanto os limites como a mão do utilizador encontram-se preenchidas com uma cor neutra. Assim que uma *track* é selecionada, tanto as mãos virtuais do utilizador como os limites mudam de cor para dar ao utilizador *feedback* visual acerca do que está a acontecer.

No início da implementação da aplicação não existiam ainda as mãos virtuais para guiar o utilizador no ecrã. Existiam, no entanto, as mesmas zonas descritas anteriormente, delimitadas pelos mesmos limites. Constatou-se então que era muito

mais difícil para o utilizador colocar a mão na zona desejada, pois tinha apenas o *feedback* da visualização e auditivo para se guiar (note-se que o *feedback* da visualização não engloba as mãos virtuais do utilizador, assunto abordado em maior detalhe na próxima secção). A partir do momento em que as mãos virtuais foram adicionadas à aplicação, tornou-se bastante mais simples a orientação do utilizador, pois o *feedback* que obtinha era relativo à localização das suas mãos no espaço virtual.

Outra consequência foi a melhoria na precisão. Antes da introdução das mãos virtuais era normal passar de uma zona para outra sem querer, obrigando o utilizador a olhar constantemente para as suas mãos reais para tentar mantê-las no mesmo sítio. Após a introdução das mãos virtuais o utilizador guia-se maioritariamente por elas e olha bastante menos para as suas mãos reais. Este facto sugeriu a possibilidade de ter ainda mais uma ou duas zonas, totalizando cinco ou seis, sem que isso se traduzisse numa perda de precisão das mãos. A hipótese, no entanto, teria de ser corroborada por testes com utilizadores efetivos. Apesar de não se ter feito este teste, os utilizadores foram consultados acerca do seu interesse em aumentar o número de zonas disponíveis na mão esquerda e os resultados serão apresentados em maior detalhe no próximo capítulo.

3.3 Interface para construção, composição de música e marcação de tempo

A interface gráfica foi construída em *HTML* e *CSS* isto é, sem recorrer a funcionalidades recentes de *HTML5* como *WebGL*, como acontece com a parte reservada à visualização. Neste capítulo serão abordadas em detalhe três partes da aplicação: a primeira está situada na parte inferior do ecrã e consiste num conjunto de áreas retangulares que representam as várias *tracks* da aplicação; em segundo lugar, uma área retangular que se encontra no canto inferior direito do ecrã, que permite regular o volume da aplicação; finalmente, na parte superior do ecrã, encontra-se um conjunto de números que representam os tempos do metrónomo.

Começando com as *tracks*, estas encontram-se dispostas de forma centrada na parte inferior do ecrã. Cada uma das áreas retangulares representa uma *track*. O

utilizador, ao selecionar um dos retângulos, pode reproduzir o som associado a essa *track*. Em cada retângulo existem dois botões, um para gravar e outro para fazer *pause* ou *play,* consoante a ação corrente. O botão de pause/play encontra-se desativado ao início pois é necessário ter algo gravado para se poder suspender a reprodução desse som. Ao carregar no botão de gravar, a cor da metade superior do retângulo muda para laranja, indicando que a aplicação se está a preparar para gravar (o início da gravação dá-se no início de cada compasso, o utilizador pode guiar-se pelos números do metrónomo que serão explicados posteriormente para saber quando é que a aplicação vai começar a gravar). Quando a gravação é iniciada, a cor *da metade superior* do retângulo muda para vermelho indicando que a gravação está a decorrer. Uma vez acabada a gravação, a cor muda para verde para indicar que a *track* está simplesmente a reproduzir o som que acabou de ser gravado. Neste momento o utilizador pode escolher parar o som da *track* que acabou de gravar (a partir deste momento o botão de pause/play torna-se clicável). Para isso basta carregar no botão de pausa, o som é parado imediatamente e a cor *da metade superior* do rectângulo muda para castanho para indicar a pausa.

Em baixo, do lado direito do ecrã, encontra-se uma área retangular com o nome *Master Volume* que consiste num *slider* que pode ser controlado pelo utilizador com o rato. A função deste *slider* é controlar o volume geral da aplicação. Não deve ser confundido com a funcionalidade associada à mão direita que também controla o volume, pois, enquanto o *slider* controla o volume geral da aplicação e, assim, de todas as *tracks*; a mão direita controla o volume individual que está a ser reproduzido da *track* que estiver naquele momento selecionada.

Finalmente, na parte superior do ecrã, de forma centrada, encontram-se dois números no formato X/X. O número da esquerda conta os tempos do metrónomo de um até quatro e está sincronizado com o som da batida da música. O segundo número conta de um até dezasseis e representa as notas que podem ser tocadas quando a mão do utilizador está na zona quatro vezes mais rápida – tabela 1. O segundo número está relacionado com o primeiro pois divide cada tempo em quatro. Na prática isto significa que, quando o primeiro número está com o valor um, o segundo vai ficar com o valor um, e depois dois, três e quatro; quando o segundo número for para o valor cinco, o primeiro toma o valor dois e assim sucessivamente.

3.4 Visualização

A componente de visualização da aplicação foi construída utilizando *WebGL*, mais especificamente, usando a biblioteca *Three.js* que abstrai muitos detalhes de implementação dos gráficos tridimensionais. Foi criada uma cena tridimensional onde existe um plano de fundo e um plano que serve de chão. Neste segundo plano estão dispostos uma série de conjuntos de blocos que representam as várias visualizações das várias tracks.

A cena criada envolve alguma complexidade em termos de computação e, para obter a melhor experiência de utilizador possível, é necessário correr a aplicação num computador que possua tanto um processador como uma placa gráfica de gama alta. O computador em que a aplicação foi produzida e testada tem a seguinte lista de *hardware*:

- *CPU – Intel Core i7 4770 3.40GHz*
- *RAM – 16Gb*
- Placa Gráfica – *GeForce 770GTX*

Num computador que tenha estes requisitos a nível de *hardware* a aplicação corre de maneira geral a 60FPS. Foi também testada a aplicação em outros computadores que tinham configurações inferiores em termos de *hardware* e os resultados foram sempre inferiores a 60 FPS.

Um dos objectivos da aplicação era conseguir a visualização clara da música criada. Visto que cada *track* toca um som único e distinto, foi também um objetivo desde o início que se conseguisse distinguir os sons de cada *track* na visualização.

Lembrando que a aplicação está dividida em dois tipos de *tracks*, as *tracks* base, que não podem ser alteradas pelo utilizador, e as *tracks* normais que podem ser manipuladas pelo utilizador, foram criados dois tipos de visualização.

3.4.1 Visualização das tracks base

Criou-se uma visualização relativa a apenas uma das *tracks* base, a que contém o som do bombo. Para este efeito, o plano vertical ao fundo do espaço tridimensional contém dois círculos posicionados nos cantos inferiores direito e esquerdo, tal como pode ser observado na figura 28, e que reagem em simultâneo, aumentando de tamanho

quando é reproduzido o som. Devido à cor que possuem, os círculos assemelham-se a uma luz pulsante. Esta visualização é iniciada assim que o som começa a ser reproduzido.

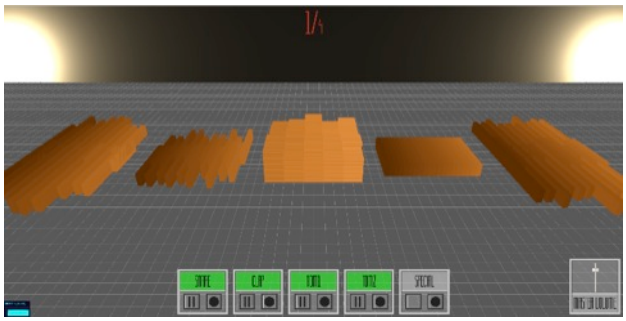


Figura 28 - Vários conjuntos de paralelepípedos a variar o seu tamanho consoante o som tocado

Os círculos presentes no plano desta visualização foram conseguidos com recurso a um *vertex shader* (Vertex Shader). Dentro do paradigma do *WebGL* existem dois conceitos importantes no que concerne a *shaders*: os *vertex shaders* e os *fragment shaders* (Fragment Shader). Ambos utilizam a informação da forma do objecto a que estão associados e transformam-na nos *pixels* que são mostrados no ecrã. Existe no entanto uma diferença entre eles, o *vertex shader* é utilizado para determinar a posição de cada um dos vértices de cada triângulo da forma geométrica a que está associado, enquanto que o *fragment shader* lida com as cores que cada triângulo tem, tendo em conta factores como luz e materiais aplicados aquele objeto. Estes *shaders* são descritos não com simples *javascript* mas utilizando uma linguagem de alto nível que foi especificamente desenhada para lidar com os requisitos próprios deste tipo de computação chamada *GLSL (OpenGL Shading Language)* (OpenGL Shading Language). Todos os *shaders*, independentemente do seu tipo, são compilados aquando da inicialização da aplicação. Se forem compilados com sucesso, então podem ser posteriormente aplicados às respectivas geometrias.

O plano vertical no fundo da cena tridimensional foi criado com recurso a ambos os *shaders*. O *vertex shader* é bastante simples e posiciona a imagem que contém os círculos de luz para a posição onde se encontra a câmara do utilizador. O *fragment shader* é já bastante mais complexo. Este *shader* foi construído a partir de um já existente (htt). O *shader* original que pode ser observado na figura 29, desenhava simplesmente um círculo de uma cor alaranjada sobre a posição do rato do utilizador.

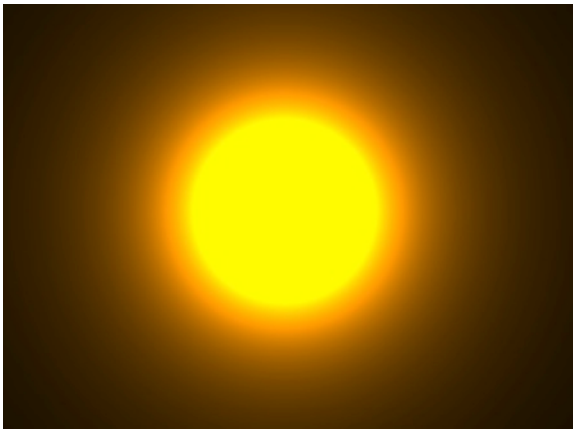


Figura 29 - Shader original

Exceto a posição do x e do y do rato do utilizador, o *shader* não possuía mais nenhum tipo de dinamismo, a sua cor não se alterava nem o tamanho do círculo. Foram feitas alterações para transformar o *shader* no estado em que se encontra presentemente. Alterou-se o código de modo a que deixasse de existir um círculo e passassem a existir dois. Esses círculos deixaram de seguir a posição do rato do utilizador e passaram a estar fixos nos cantos inferiores do plano. A cor também foi mudada de modo a ter tons mais amarelos e brancos e menos laranja. No *shader* original (Figura 29) é possível ver que o círculo é composto por três partes, um círculo interior que possui uma cor mais amarela, uma coroa exterior a esse círculo que é mais alaranjada e finalmente um gradiente de luz que vai do laranja ao preto, que é a cor do fundo da imagem. Na versão da aplicação, a coroa foi encurtada de modo a dar mais importância ao círculo interior e ao gradiente.

3.4.2 Visualização das *tracks* normais

Relativamente às *tracks* normais, que podem ser manipuladas pelo utilizador, existem presentemente cinco, podendo no entanto ser adicionadas mais no futuro. A cada uma corresponde um dos blocos de paralelepípedos que estão dispostos horizontalmente, tal como se pode ver na figura 28.

Os blocos são animados quando é reproduzido o som da respectiva *track* e o seu movimento é função da média das frequências que compõem o som. Para facilitar a identificação da *track* que está a ser reproduzida definiram-se três tipos de animações:

uma para os blocos dos extremos, outra para o segundo e quarto blocos e uma terceira para o bloco do meio.

Os blocos dos extremos são compostos por 20 paralelepípedos, com a dimensão maior paralela ao eixo dos *zz*, e efetuam uma mudança de escala segundo esse eixo. O bloco do meio é constituído por um conjunto de 5x5 paralelepípedos e efetua uma mudança de escala segundo o eixo dos *yy*. Finalmente, os restantes dois conjuntos de blocos, igualmente constituídos por 20 paralelepípedos, dispostos também com a dimensão maior segundo o eixo dos *zz*, efetuam uma translação segundo este eixo de forma desencontrada, ou seja, se o primeiro paralelepípedo efetua uma translação positiva no eixo dos *zz*, o segundo efetua uma translação negativa, e assim sucessivamente. Em todas as visualizações existe sempre alguma aleatoriedade no tamanho dos paralelepípedos quando reagem ao som reproduzido.

Por último, uma palavra relativamente ao elemento gráfico que se encontra no canto inferior esquerdo do ecrã. Este pequeno retângulo mede os FPS da aplicação com um gráfico azul. Se o utilizador carregar nele, o gráfico muda de azul para verde e mostra quantos milissegundos é que demora entre cada *frame*. Este elemento foi incluído na aplicação intencionalmente, por duas razões. Em primeiro lugar, quando a aplicação estava a ser desenvolvida era importante ter esta métrica sempre presente para tomar decisões relativamente a opções de implementação devido ao desempenho. Em segundo, porque a aplicação construída é exigente em termos de *hardware*, e, por isso, os utilizadores da aplicação têm algo que lhes mostra se ela está a correr nas melhores condições possíveis ou não (o ideal seria que a aplicação corresse a 60 fps).

3.5 Sumário

Apresentou-se neste capítulo as principais componentes que constituem a aplicação criada. Explicou-se em detalhe cada uma delas e como foram construídas e desenhadas.

No capítulo seguinte são apresentados os vários testes que foram realizados para validar as escolhas feitas aquando da implementação. Tentou-se avaliar tanto a usabilidade como a utilidade da aplicação quer por utilizadores comuns como por peritos nas áreas musical e de interação.

Capítulo 4

Testes com utilizadores

Neste capítulo são apresentados os resultados dos testes com utilizadores que foram efectuados.

4.1 Metodologia

Na fase final do projeto efetuaram-se uma série de testes com utilizadores. Consideraram-se dois conjuntos de utilizadores, peritos e utilizadores comuns, sendo que os peritos se dividiram por sua vez em duas áreas: peritos na área musical e peritos na área de interação. A entrevista decorreu da seguinte forma: em primeiro lugar era dada uma breve explicação acerca do funcionamento da aplicação, de seguida o utilizador experimentava a aplicação por si próprio e finalmente respondia a um conjunto de perguntas. O inquérito pode ser consultado no anexo A. As perguntas contidas no inquérito pretendiam, por um lado, obter *feedback* dos participantes acerca da usabilidade e qualidade da aplicação desenvolvida e, por outro lado, recolher sugestões e opiniões para melhorias futuras.

Para além das perguntas iniciais, que foram feitas para definir um perfil de utilizador, as restantes incidiram sobre três aspetos principais: áudio, interação e visualização. Quanto ao primeiro, quis-se avaliar a pertinência da escolha dos sons e a facilidade do processo de gravação; o segundo incidiu sobre a interação do utilizador com o dispositivo *Leap Motion* e tentou-se apurar o quão fácil e natural era esta interação; finalmente, no terceiro, avaliou-se a visualização dos sons no espaço tridimensional e se existia de facto uma correspondência óbvia e clara entre os sons reproduzidos e as respetivas visualizações.

O facto de terem sido feitas entrevistas a utilizadores peritos, além dos comuns, revelou-se uma considerável vantagem, especialmente no que se refere a peritos na área musical. Uma vez que a aplicação desenvolvida se insere claramente na categoria musical, foi interessante ter entrevistado peritos dessa área que tinham diversos *backgrounds*, e que, portanto, puderam fornecer perspectivas e opiniões diferentes acerca dos vários aspetos da aplicação.

As entrevistas efetuadas tiveram uma duração média de 30 minutos com cada utilizador comum e de cerca de uma hora com cada perito. As perguntas feitas tanto aos utilizadores comuns como aos peritos eram iguais.

As perguntas efectuadas tinham três formatos distintos:

- Perguntas de classificação, onde existia uma escala de 1 a 5 e era pedido ao utilizador que classificasse uma funcionalidade ou aspeto da aplicação.
- Perguntas de alternativa, onde eram apresentadas duas soluções para uma funcionalidade ou aspeto da aplicação e era pedido ao utilizador que indicasse aquela que preferia.
- Perguntas de alternativa semi-abertas, onde era pedido ao utilizador que escolhesse uma de duas propostas, sendo que uma delas poderia ser sugerida pelo utilizador.

É de notar que em alguns casos o utilizador não pôde experimentar a aplicação num computador que oferecesse a melhor experiência de utilização possível. Tendo este facto em conta, foi dada uma classificação de 1 (menor) a 5 (maior) ao computador em que o utilizador experimentava a aplicação e os resultados foram analisados tendo isto em conta.

4.2 Perfil dos utilizadores

Os testes foram realizados sobre um universo de 23 utilizadores com idades entre os 17 e 50 anos. Os inquéritos foram respondidos de forma anónima. Para definir o perfil de cada utilizador, para além de serem colocadas perguntas sobre a idade e sobre o sexo, pediu-se também que o utilizador respondesse a três perguntas de auto-avaliação: se costuma jogar jogos de computador ou de consolas, se sabe tocar algum instrumento musical e, finalmente, qual é a sua mão dominante, esquerda, direita ou

ambas. Com as respostas a estas perguntas pretende identificar-se qual o nível de experiência que o utilizador tem com aplicações interativas deste tipo e também, visto que se trata de uma aplicação de cariz musical, qual o seu nível de experiência nessa área (figura 30). A pergunta relativa à mão dominante está relacionada com uma outra pergunta que é colocada na secção de interação e que será abordada posteriormente (figura 31).

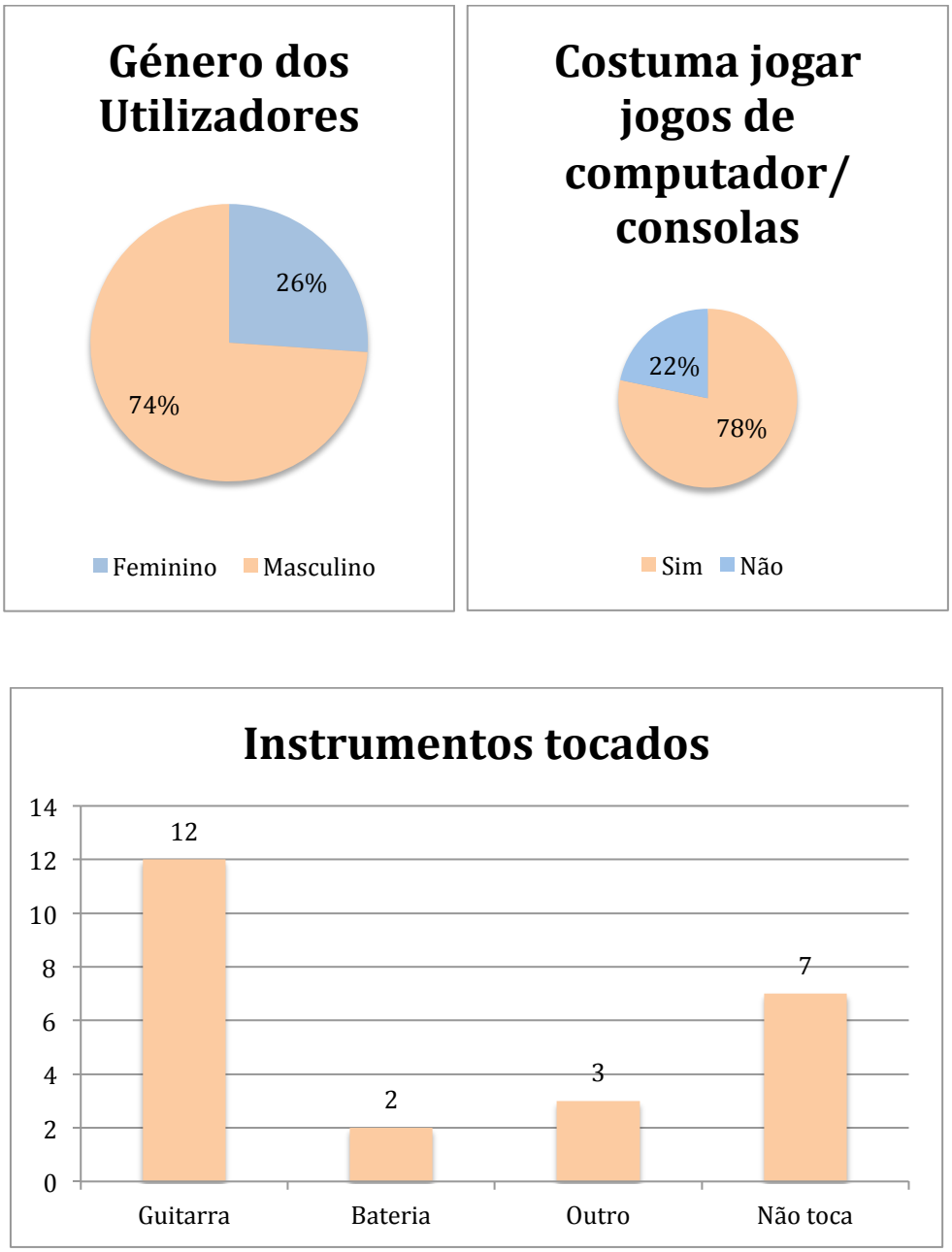


Figura 30 - Gráficos da distribuição dos utilizadores de teste por género, por uso de computadores ou consolas para jogar jogos interativos e por instrumentos que tocam.

Foi possível reunir um conjunto de pessoas que tinham experiência tanto como instrumentistas (77% dos inquiridos sabia tocar pelo menos um instrumento) como a jogar jogos de computador ou de consolas (nos quais 78% tinham alguma experiência) – figura 30.

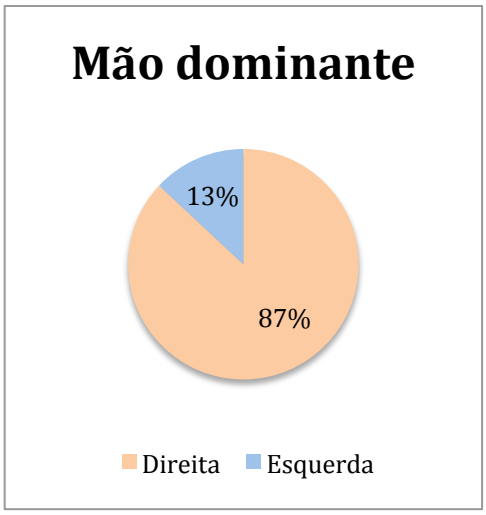


Figura 31 - Gráfico da distribuição da mão dominante dos participantes

Dentro do universo de pessoas entrevistadas existiam algumas que, devido ao seu *background* e formação, foram considerados peritos nas áreas musical e de interação. A opinião destas pessoas tem naturalmente um peso maior, nas áreas em que são especialistas, do que os utilizadores comuns, pois apesar de a aplicação ter como publico-alvo alguém que não tem qualquer conhecimento musical, estas pessoas puderam fazer sugestões relativamente a pormenores musicais e de interação que beneficiariam o utilizador comum. Relativamente aos peritos na área musical, estes não só sabem tocar um instrumento com um nível avançado, como têm conhecimentos musicais aprofundados, ou são músicos profissionais ou que sabem trabalhar em produção musical. Em relação aos peritos em interação foram consideradas pessoas que quer devido à atividade profissional, quer devido aos projetos desenvolvidos a nível académico, lidam com interfaces regularmente e têm prática em criar boas experiências de utilização para o utilizador. Quatro pessoas foram consideradas como sendo peritos na área musical e duas como peritos na área de interação.

4.3 Análise dos resultados

De seguida são apresentadas as conclusões dos resultados obtidos pelas entrevistas que foram conduzidas juntos dos utilizadores. São também apresentadas as principais opiniões e comentários que os utilizadores expressaram quando utilizaram a aplicação. As três próximas secções seguem a estrutura utilizada no inquérito: áudio, interação e visualização.

É de referir no entanto o seguinte, os testes feitos aos utilizadores, por impossibilidades práticas, foram realizados em diferentes computadores que possuíam naturalmente hardware diferente. Em cada entrevista feita a cada utilizador foi dada uma nota de um a cinco (sendo um um computador com hardware menos bom bom e cinco um computador com o hardware onde a aplicação corre de forma ideal) e os resultados das várias questões foram analisados tendo este facto em conta. Existem várias perguntas em que o desempenho do computador onde o utilizador experimentou a aplicação podia ser condicionante da resposta. No entanto isto não se revelou tão problemático quanto se pensava. Em primeiro lugar, porque o número de entrevistas que foram realizadas num computador que não fosse ideal foram apenas quatro; em segundo lugar, porque mesmo nestes casos houve apenas um número reduzido de respostas que saíram da norma das respostas dadas pelos utilizadores que fizeram os testes num computador ideal. Finalmente, as respostas que saíram da norma e que podem influenciar o resultado final são devidamente referidas.

4.3.1 Áudio

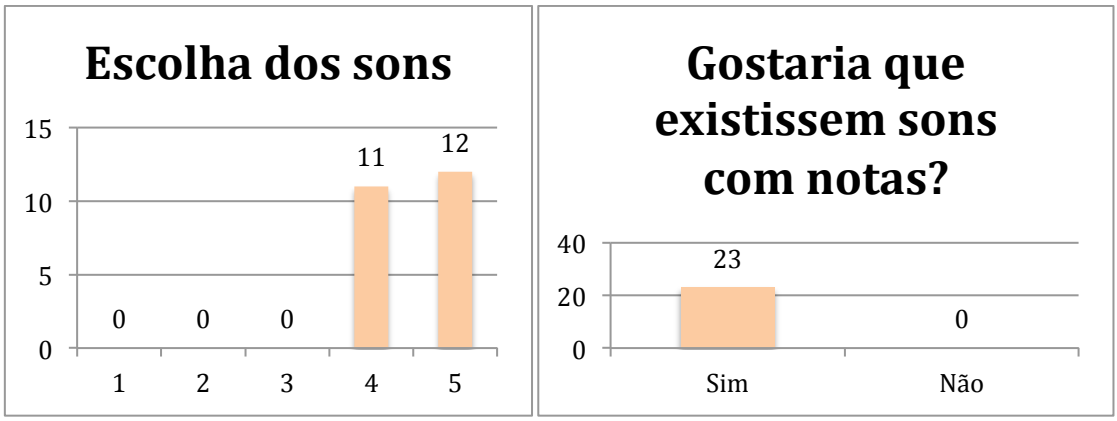


Figura 32 - Distribuição das respostas dos participantes relativas à pertinência dos sons escolhidos para a aplicação e relativamente à possibilidade de existirem sons não apenas de ritmo.

Os entrevistados revelaram-se contentes com a escolha dos sons de ritmo que estavam presentes na aplicação (100% igual ou superior a 4 e 52% com nota igual a 5). Quando foi perguntado se gostariam de experimentar também sons com notas e não apenas de ritmo a resposta foi unânime (100% disseram que sim) – figura 32.

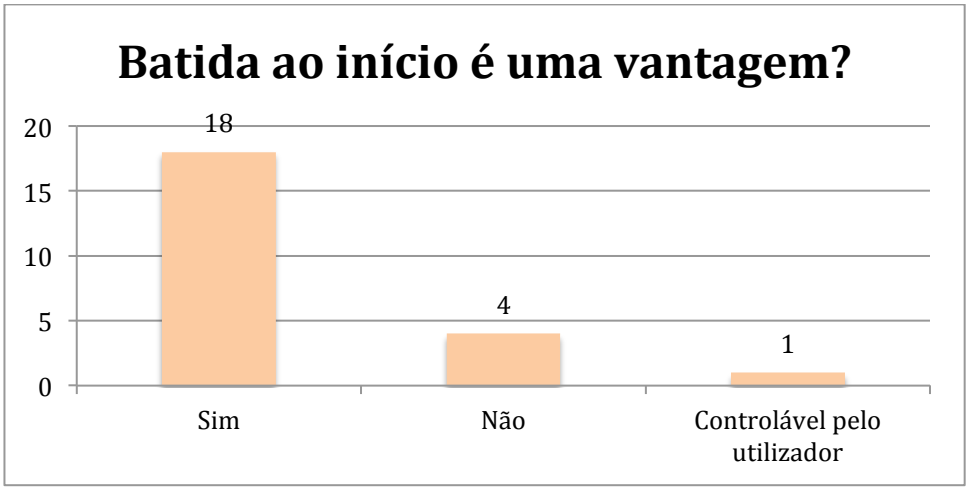


Figura 33 - Distribuição das respostas dos participantes relativamente à presença da batida na aplicação

Para além dos sons que podiam ser controlados pelo utilizador, perguntou-se também acerca do som da batida da *track* base. A suposição ao início era que esta servisse como ajuda à composição, pelo menos aos utilizadores que não tivessem grande prática musical. Esta suposição veio a ser confirmada com os resultados dos inquéritos com 78% das respostas a indicarem que a batida ao início oferece uma vantagem, pois dá uma base sobre a qual trabalhar. Foram no sentido negativo 18% das respostas e houve ainda um entrevistado que sugeriu que o tocar da batida desde o início deveria ser uma funcionalidade ajustável pelo utilizador (figura 33). Vale a pena referir que, dentro do universo das respostas negativas (apenas quatro), duas dessas respostas foram dadas por peritos. Este facto é importante, pois os peritos tendem a uma utilização mais criativa da aplicação e podem dispensar melhor esta espécie de "muleta". Apesar de ser de cariz musical, a aplicação foi pensada para ser usada por qualquer pessoa e esta funcionalidade foi adicionada especificamente para o utilizador comum. Numa iteração seguinte seria de facto interessante existir uma forma de ligar ou desligar a batida automática para que a aplicação se adeqúe melhor aos dois tipos de

utilizadores. Note-se que a opção da batida ajustável não existia no inquérito mas achou-se relevante considerar este valor na fase de tratamento de dados.

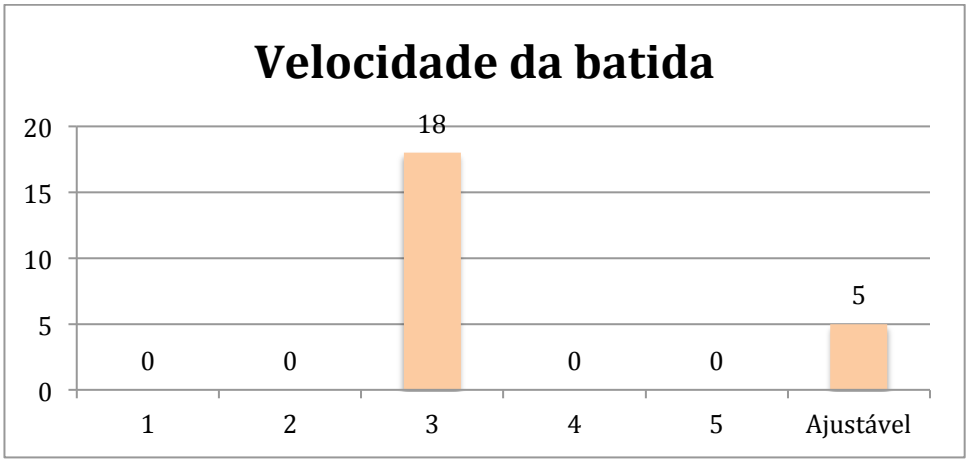


Figura 34 - Distribuição das respostas dos utilizadores teste relativamente à velocidade da batida da música

Ainda quanto à batida, foi perguntado acerca da sua velocidade. A batida estava definida para um BPM (*beats per minute*) de 90 e quis-se saber se esta velocidade promovia a fácil criação de música ou se, pelo contrário, estava demasiado rápida ou demasiado lenta. Para esta resposta foi definida uma escala diferente da escala usada nas outras perguntas. Deu-se-lhe valores de 1 a 5 mas definiu-se que estes significavam o seguinte: 1 – Velocidade da batida deveria ser muito mais lenta; 2 – Velocidade da batida deveria ser mais lenta; 3 – Velocidade da batida ideal; 4 – Velocidade da batida deveria ser mais rápida; 5 – Velocidade da batida deveria ser muito mais rápida (figura 34). Tendo em conta esta escala, esperar-se-ia que os valores se aproximassem do 3, em vez de se aproximarem do 5, como seria de esperar nas outras perguntas. Foi, de facto, isto que aconteceu, sendo que 75% dos entrevistados responderam com o valor 3 a esta pergunta. Os restantes, à semelhança do que aconteceu com a pergunta anterior, deram uma resposta que não se encontrava no questionário, responderam que a velocidade da batida deveria se regulável. Entendeu-se que esta resposta, apesar de não estar originalmente no questionário, deveria ser considerada quando fosse feito o tratamento dos dados, e assim aconteceu. É também de notar que, como na pergunta anterior, dos quatro utilizadores a darem esta resposta dois eram peritos. Este facto segue também a linha da resposta anterior, revelando consistentemente que os utilizadores peritos, com mais experiência musical, gostam de ter mais controlo sobre os vários elementos que constituem a secção musical da aplicação.

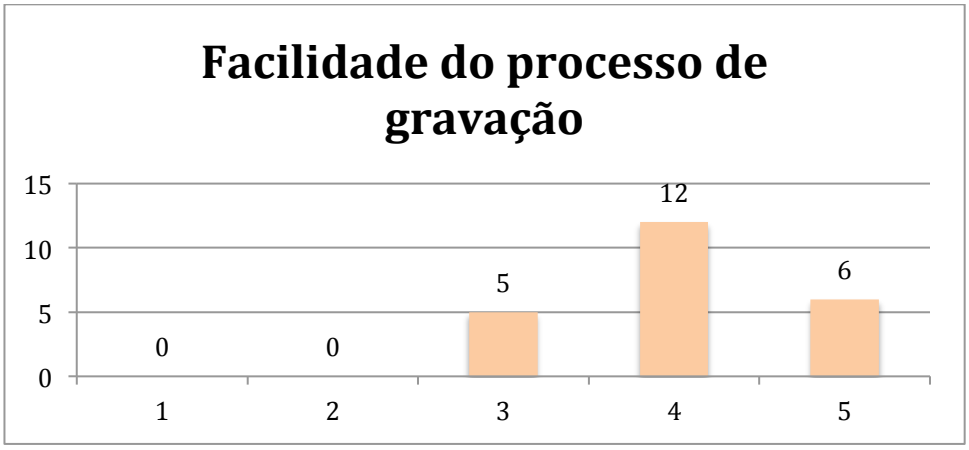


Figura 35 - Distribuição das respostas dos participantes relativamente à facilidade do processo de gravação

Finalmente, a última funcionalidade de áudio que foi testada com utilizadores foi o processo de gravação. Com 78% dos utilizadores a registarem uma resposta com o valor igual ou superior a 4 este foi o aspeto que foi menos conseguido na secção áudio do inquérito (figura 35). Para testar esta funcionalidade foi pedido aos utilizadores que experimentassem gravar algumas *tracks*. Ao princípio os utilizadores não percebiam imediatamente como funcionava o processo de gravação, devido em parte ao facto de este só começar quando fosse altura do primeiro tempo e não quando os utilizadores carregavam no botão (isto, presentemente, não é intuitivo). Apesar da confusão inicial dos utilizadores, passadas algumas tentativas, de maneira geral, conseguiam perceber como é que o processo funcionava e não acharam demasiado complexo.

De uma maneira geral, relativamente ao áudio, a resposta dos utilizadores foi positiva. Para trabalho futuro seria interessante incorporar sons não apenas de ritmo mas que tenham notas. O processo de gravação é de facto um aspeto que tem de ser melhorado, não mudando o processo em si, mas criando mecanismos que permitam ao utilizador perceber imediatamente como é que ele funciona.

Para conseguir isto poderia ser criada uma mensagem que avisasse o utilizador de que, dentro de um determinado número de tempos, a aplicação iria começar a gravar. A mensagem deveria também permanecer enquanto a aplicação estivesse a gravar e deveria informar quando terminasse. Seria também interessante criar uma maneira visual diferente de representar os tempos que regem a aplicação. Poderia, por exemplo, ser criada uma *timeline* horizontal que tivesse as marcas dos 4 tempos (figura 36) e um tipo de elemento gráfico que percorresse essa *timeline* da esquerda para a direita para

indicar o momento exacto em que a aplicação se situa relativamente ao metrónomo. Isto substituiria a função que os números no topo do ecrã desempenham neste momento.

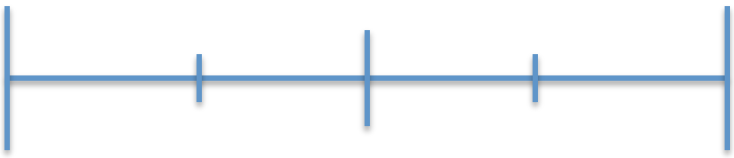


Figura 36 - Exemplo de um elemento gráfico que substituísse os números do metrónomo

4.3.2 Interação

A principal intenção das perguntas feitas relativamente à interação foi saber a opinião dos utilizadores sobre o uso das mãos para reproduzir som, sobre a posição das mãos no espaço tridimensional e sobre as funcionalidades que controlam.

Relativamente às zonas das mãos dos utilizadores quis-se saber em primeiro lugar se o número de zonas que existem atualmente na aplicação permite ao utilizador ter um controlo preciso sobre o som que está a reproduzir.

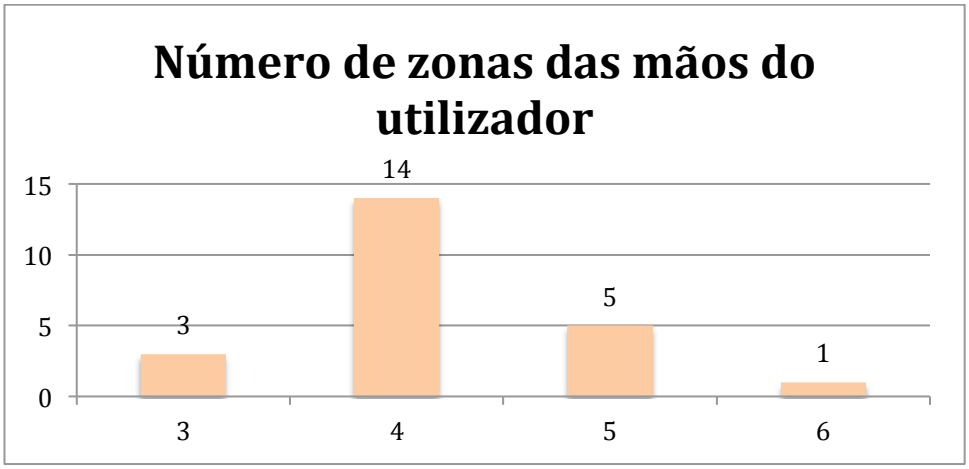


Figura 37 - Distribuição das respostas dos participantes relativamente ao número ideal de zonas das mãos do utilizador

A pergunta feita aos entrevistados tinha duas opções, uma sobre a satisfação com as quatro zonas atualmente existentes e outra onde se pedia para o utilizador especificar um número de zonas que considerasse ótimo. Confirmaram as quatro zonas existentes como ideal 61% dos entrevistados. Preferiram cinco ou seis zonas 26%. Finalmente, 13% escolheram três (figura 37). Estes valores sugerem que o número de zonas possa

ser aumentado para pelo menos cinco zonas sem que isso constitua uma perda significativa em termos de precisão na zona em que se encontra a mão do utilizador.

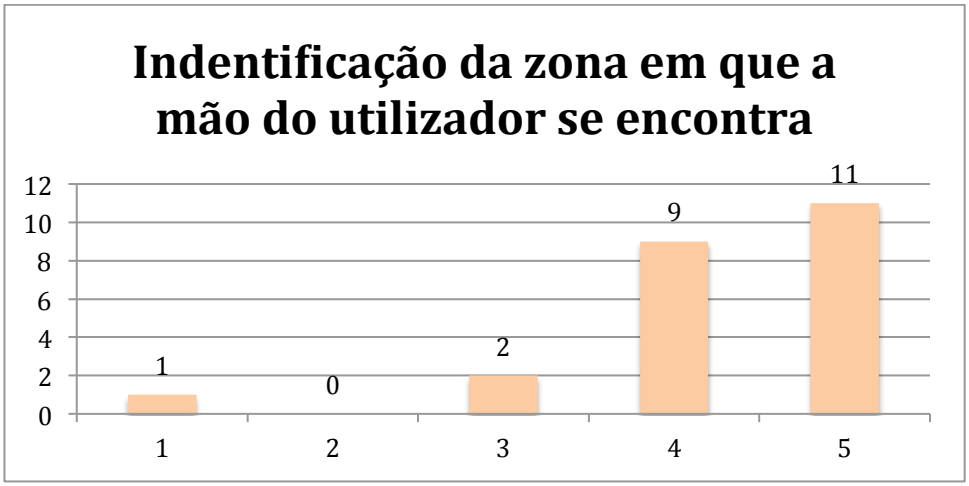


Figura 38 - Distribuição das respostas dos participantes relativamente à facilidade de identificação da zona em que a mão do utilizador se encontra

A facilidade de identificar exatamente em que zona se encontravam as mãos dos utilizadores teve 87% de notas com o valor quatro ou superior (figura 38). Um dos utilizadores deu a nota um a esta pergunta, no entanto isto deveu-se maioritariamente à qualidade do computador onde foi experimentada a aplicação. Por impossibilidade de arranjar um computador onde a aplicação corresse com maior fluidez acabou por se usar um sistema fraco no que toca a desempenho.

Independentemente das notas dadas pelos utilizadores, uma das opiniões que foi quase unanimemente expressa foi o facto de a transparência dos planos horizontais que delimitavam as várias zonas estar demasiado acentuada e, por isso, não estar tão visível quanto deveria. Houve também uma sugestão no sentido de se criar volumes e não apenas planos para delimitar as várias zonas de interação.

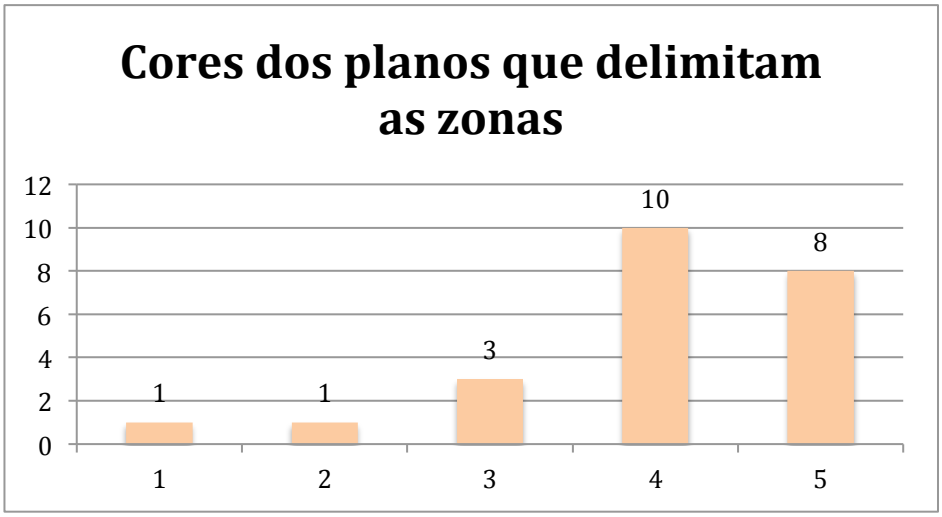


Figura 39 - Distribuição das respostas dos utilizadores de teste relativamente à pertinência das cores usadas para os planos que delimitam as várias zonas de ação das mãos do utilizador

79% dos utilizadores deram uma nota igual ou superior a 4 quando lhes foi perguntado se achavam uma boa escolha as cores cinzento e verde para colorir os planos que delimitavam onde as mãos do utilizador se encontravam – figura 39.

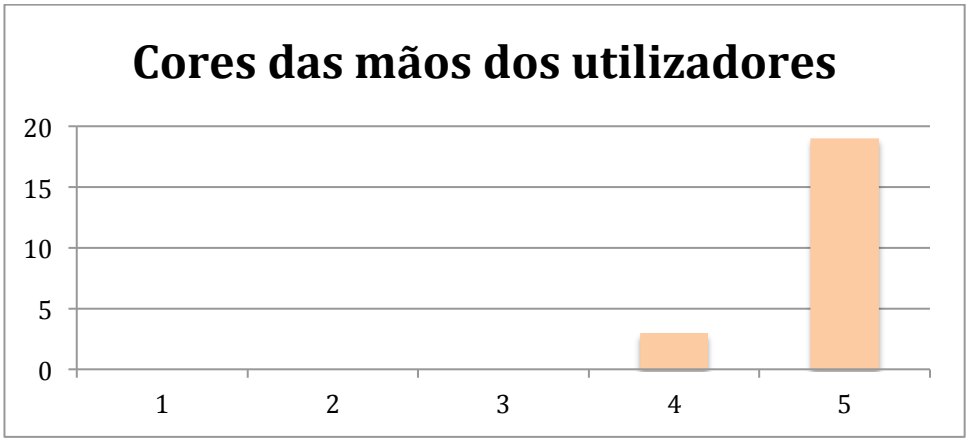


Figura 40 - Distribuição das respostas dadas pelos participantes relativamente à pertinência da escolha das cores usadas para colorir as mãos virtuais do utilizador

Relativamente às cores azul e vermelho que as mãos virtuais dos utilizadores tomavam, foi perguntado acerca da pertinência dessa escolha. Aqui os resultados foram melhores do que na pergunta anterior, visto que 86% dos inquiridos deram a nota máxima (figura 40).

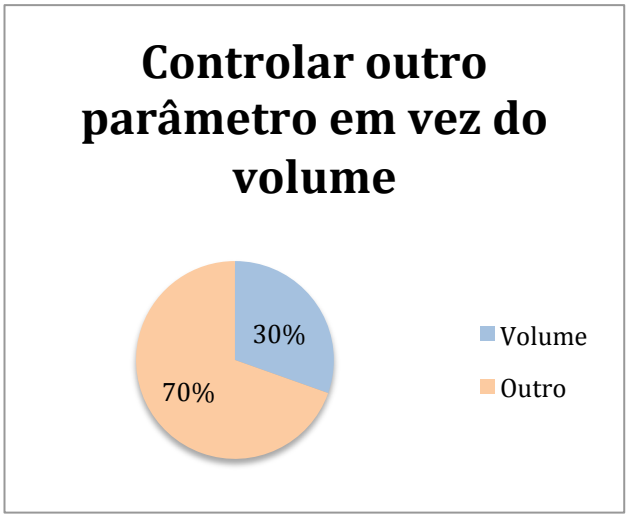


Figura 41 – Distribuição das respostas dos participantes relativamente à escolha do volume como parâmetro a ser controlado pela mão direita do utilizador.

A pergunta acerca da funcionalidade controlada pela mão direita foi feita com a intenção de determinar qual seria a melhor função, permanecer a controlar o volume do som ou controlar outro parâmetro como, por exemplo, a distorção do som tocado. Com 70% dos utilizadores a preferirem controlar outro parâmetro, esta seria uma funcionalidade que seria interessante implementar numa futura iteração da aplicação (figura 41).

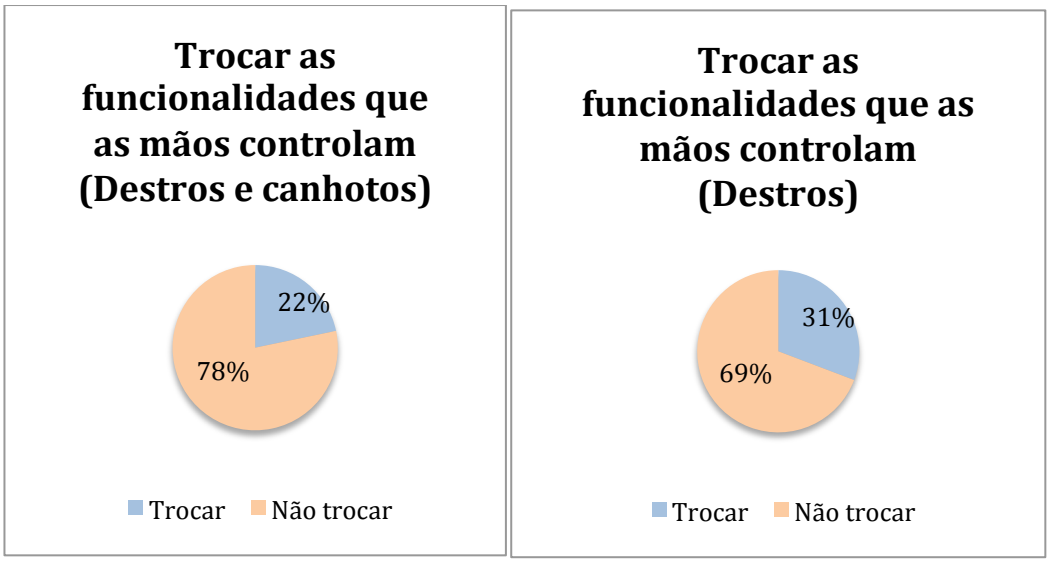


Figura 42 – Distribuição das respostas dos utilizadores teste relativamente a trocar ou não as funcionalidades de cada mão controla. Respostas dadas por participantes destros e canhotos e apenas por destros.

A pergunta relativa a trocar as funcionalidades controladas por cada mão do utilizador é a razão pela qual, ao traçar o perfil de utilizador, foi perguntado qual a mão dominante do entrevistado. Queria-se retirar conclusões relativamente à utilização da

aplicação por parte de destros e de canhotos. Infelizmente apenas se conseguiu entrevistar 3 pessoas que eram canhotas e portanto quaisquer conclusões são frágeis. Ainda assim, os resultados foram analisados tendo em conta o conjunto de todos os utilizadores e considerando apenas destros e canhotos separadamente.

Quando é tido em conta o universo de todos os utilizadores temos 22% destes que preferiam trocar as funcionalidades controladas por cada mão. No entanto, quando analisamos apenas o universo dos utilizadores destros esta percentagem aumenta para os 31% (figura 42). Relativamente aos canhotos, temos 100% destes a escolherem não trocar. Estes dados, embora assentes num universo diminuto de utilizadores, sugere que para os utilizadores destros deveria existir pelo menos a possibilidade de trocar as funcionalidades das mãos e fazer o volume variar com a mão esquerda e o momento em que o som é tocado com a mão direita. Ainda assim vale a pena referir o seguinte: quando esta pergunta foi feita aos utilizadores, a maior parte deles, independentemente da resposta que deu e também independentemente de ser canhoto ou destro, não deu muita importância ao que cada mão fazia. Isto deve-se provavelmente ao facto de esta ser uma interface nova para os utilizadores e ser mais fácil a habituação inicial.

4.3.3 Visualização

Quis-se averiguar principalmente se as visualizações implementadas para os vários sons tiveram o efeito pretendido e se eram facilmente associáveis aos sons que estavam a tocar naquele momento.

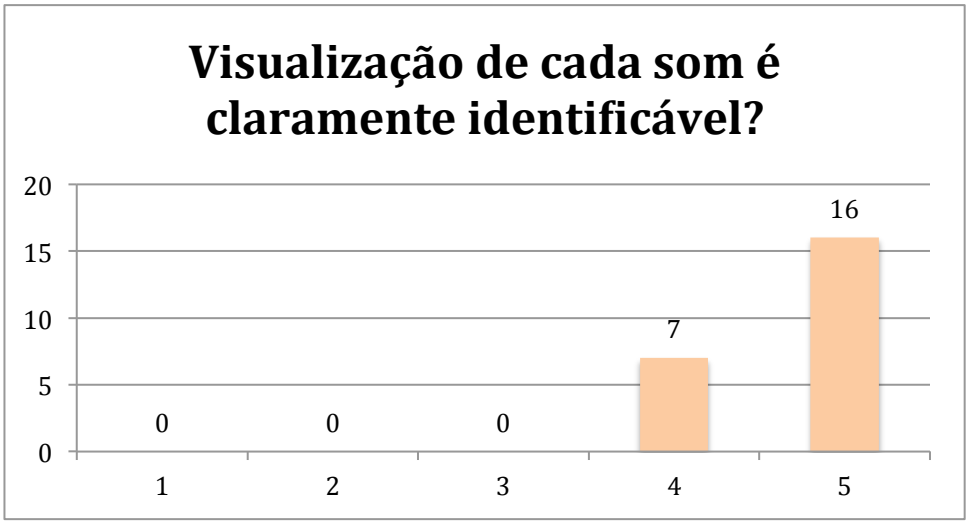


Figura 43 - Distribuição das respostas dos utilizadores relativamente à facilidade de identificação da visualização de cada som

A questão relativa à clara identificação da visualização de cada som é sem duvida a pergunta mais importante. A intenção desde o início sempre foi que o utilizador associasse de maneira fácil e natural cada som reproduzido a cada visualização. Tal como foi referido no capítulo do trabalho relacionado, à medida que a complexidade das visualizações (especialmente as tridimensionais) aumenta, torna-se também cada vez mais difícil distinguir cada som de maneira visual.

A resposta foi bastante positiva, com 70% dos entrevistados a darem a nota 4 e os restantes 30% a darem a nota máxima (figura 43).

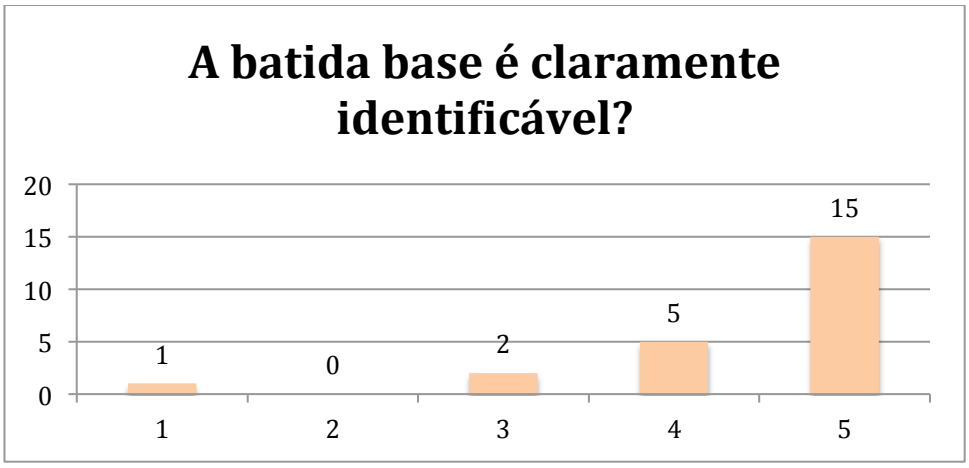


Figura 44 – Distribuição das respostas dos utilizadores relativamente à facilidade de identificação da visualização da batida base.

Com menos importância que a pergunta anterior, mas ainda assim relevante, foi a pergunta acerca de quão fácil era identificar a visualização da batida base. Deram nota máxima a esta pergunta 65% dos utilizadores (figura 44). É também certo que houve notas com valor um e três, mas, especialmente na situação em que o utilizador deu a nota um, foi um caso, mais uma vez, em que não foi possível arranjar um computador de melhor qualidade onde fazer os testes.

Alguns dos utilizadores não se aperceberam de que a luz pulsante presente no fundo da cena tridimensional era, de facto, a visualização da batida base. Por um lado, isto revela que a visualização não é intrusiva, o que, em parte, segue o propósito para o qual foi adicionada a batida base: estar presente e ajudar o utilizador a construir música mas não ser intrusiva e não se sobrepor às *tracks* principais.

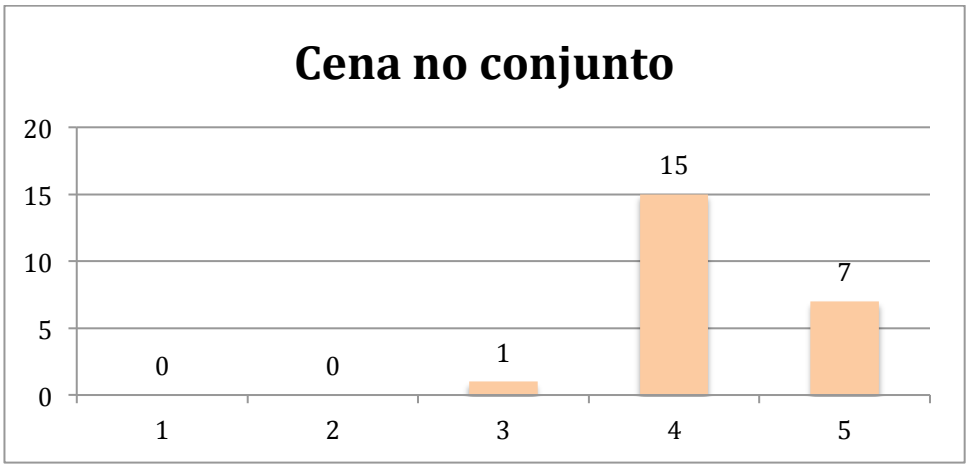


Figura 45 – Distribuição das respostas dos utilizadores teste relativamente à sua opinião da cena no seu conjunto.

Independentemente das visualizações, perguntou-se aos utilizadores o que achavam da cena tridimensional no seu conjunto, se esta foi bem conseguida e se produzia um resultado agradável do ponto de vista gráfico. Houve 65% de utilizadores que avaliaram com nota 4 este aspeto da aplicação e 31% deram a nota 5 (figura 45). Foi importante fazer esta pergunta, pois a aplicação foi construída desde o início com o intuito de ser agradável esteticamente.

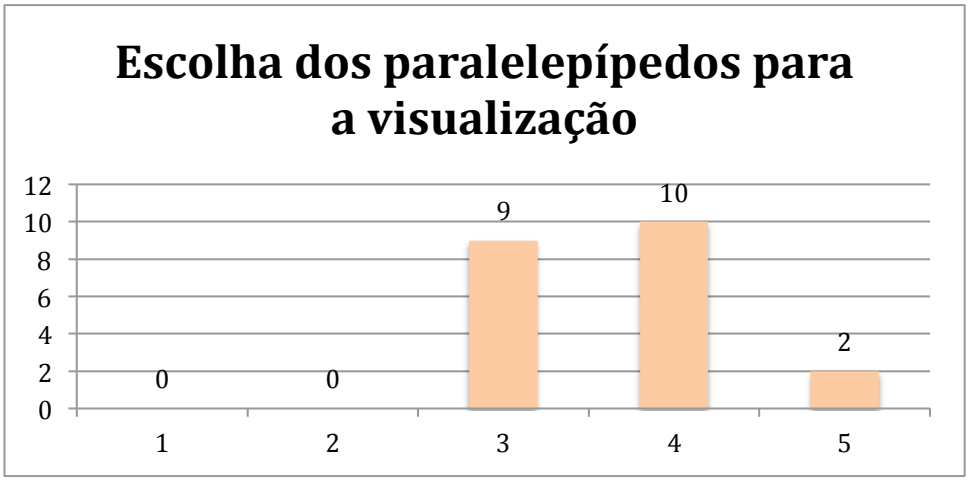


Figura 46 - Distribuição das repostas dos utilizadores de teste relativamente à escolha dos paralelepípedos como visualização

Foi perguntado aos entrevistados o que achavam da escolha dos paralelepípedos para a visualização das diferentes *tracks*. Consideraram-na escolha pertinente 43%, que deram a nota 3, e 48% consideraram uma boa escolha ao dar a nota 4 (figura 46). Além disto, foi perguntado se os utilizadores tinham alguma sugestão para outras visualizações. A maior parte não deu qualquer sugestão pois não imaginavam nenhuma. Dos utilizadores que deram, de facto, sugestões, estas assentavam principalmente em usar outras formas geométricas, em vez de paralelepípedos, para representar cada uma das *tracks*. Estas formas consistiam em cubos, prismas, pirâmides mas também formas curvas como esferas ou cilindros. Mas nenhuma sugestão fugia das formas geométricas. Para além disto, foram também sugeridas outras formas de animação, que também não se afastavam do que a aplicação oferece: translações, rotações e mudanças de escala.

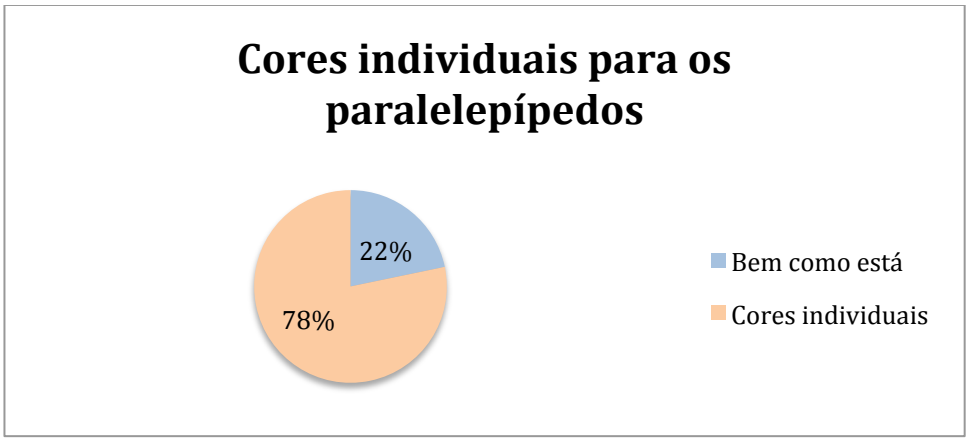


Figura 47 - Distribuição das respostas dos participantes relativamente às cores dos paralelepípedos

Por último foi perguntado aos utilizadores acerca das cores que os paralelepípedos tinham e o que seria melhor, dar a cada conjunto de paralelepípedos uma cor individual que fizesse mais fácil a identificação, ou se, por outro lado, seria melhor manter as cores como estavam. 78% dos utilizadores afirmaram que cores individuais seria melhor para distinguir os vários conjuntos de paralelepípedos e os restantes 22% disseram que as cores estavam bem como estavam (figura 47). Apesar destas respostas, também afirmaram que a ordem em que os conjuntos de blocos estão permite facilmente distingui-los, sendo a cor individual apenas um elemento adicional e não imprescindível.

Capítulo 5

Conclusões e trabalho futuro

Desenvolveu-se uma aplicação interativa para a criação de música. A aplicação tem um objetivo lúdico e permite aos utilizadores tocar e compor diversos sons de modo a formar uma música através da interação com o dispositivo Leap Motion. Ao mesmo tempo que os sons são reproduzidos é deles apresentada uma visualização tridimensional. Esta aplicação combina três componentes fundamentais: áudio, para reprodução, gravação e combinação de sons; interação, onde é usado o dispositivo Leap Motion para tocar os sons; e a visualização, onde são usados gráficos tridimensionais para visualizar a música criada. A aplicação foi construída com base nas tecnologias *HTML5, CSS3* e *javascript,* usando extensivamente *WebGL* e *Web Audio API*, especificamente para o browser *Google Chrome* em ambiente *Windows*. Conseguiu-se um bom nível de desempenho que permite a gravação e reprodução de som de forma precisa e em tempo-real.

O efeito final obtido com a aplicação e particularmente com a junção dos três módulos discutidos anteriormente (áudio, interação com o dispositivo *Leap Motion* e visualização) foi testado com utilizadores comuns e também com especialistas nestas áreas, a fim de obter *feedback* acerca dos vários aspetos que constituem a aplicação. Esta possui um conjunto básico de funcionalidades que demonstram a potencialidade de uma aplicação deste género. O interesse demonstrado pelo carácter inovador da aplicação por parte dos utilizadores manifesta, que existe um crescente interesse por novas formas de interação especialmente, no que toca a aplicações de cariz musical. Muitos deles revelaram uma grande vontade na continuação do desenvolvimento da aplicação, tendo inclusive sugerido funcionalidades que gostariam de ver concretizadas.

Embora a vertente de áudio e de visualização não sejam por si só particularmente inovadoras, a interação com base no dispositivo Leap Motion é-o, bem como a junção destas três vertentes numa única aplicação.

A aplicação desenvolvida contém um conjunto mínimo de funcionalidades, e foi sempre pensada para servir como base e para lhe serem adicionadas novas funções. Durante a realização do projeto foram surgindo novas ideias para o seu desenvolvimento que, por manifesta insuficiência de tempo, não puderam ser implementadas e foram sendo relegadas para segundo plano. Este conjunto de ideias foi alargado a partir das opiniões que os utilizadores exprimiram quando foram feitas as entrevistas.

Em relação ao trabalho futuro foram consideradas três áreas: áudio, interface e visualização. Relativamente à parte de áudio da aplicação, tencionou-se sempre juntar não só mais sons de ritmo como também sons que possuíssem notas. Dentro do universo dos sons com notas poderia ser feita a distinção entre sons discretos (como um xilofone) e contínuos (como um violino). A utilização deste tipo de sons levantará novos problemas, não só em termos de áudio e do seu processamento (mais desafiante nos sons contínuos do que nos discretos) como também de interação. Como interagir com sons que têm várias notas? Para alguém que tenha *background* musical é interessante (como aliás foi referido por vários peritos aquando das entrevistas) ter um nível grande de controlo sobre as notas, quer quanto à sua quantidade, quer quanto ao momento em que é tocada.

Relativamente aos sons contínuos, afigura-se aconselhável o recurso a sons sintetizados em vez de sons reais, tal como é feito com a maior parte dos sons usados nesta aplicação. Deste modo será mais fácil a mudança de nota sem a percepção pela parte do utilizador de qualquer problema, como cortes abruptos. A menos que seja utilizado algum tipo de biblioteca ou *framework* que faça este trabalho pelo programador, isto não é algo trivial, implicando mesmo algum conhecimento aprofundado em termos musicais.

Ainda dentro da parte de áudio, o processo de gravação e reprodução pode sofrer também algumas alterações. Com base nas opiniões expressadas pelos utilizadores e peritos, pensou-se numa nova forma de apresentar esta funcionalidade. Concretamente, a etapa do processo de gravação em que a aplicação se prepara para gravar, porque o metrónomo ainda não chegou ao tempo um, deveria ser mudada. A mudança que se

propõe é a seguinte: quando o utilizador carrega no botão de gravar, a aplicação mostraria um aviso textual como "A PREPARAR PARA GRAVAR EM 4". O número iria decrescendo com base no tempo em que o metrónomo se encontrasse naquele momento. O início da gravação continuaria sempre a dar-se no primeiro tempo. Com este aviso a ser mostrado e fazendo com que ele fosse mostrado no mínimo 4 tempos antes de a aplicação começar, de facto, a gravar, pode antecipar-se uma maior facilidade de utilização.

As mudanças relativas aos sons com notas e ao processo de gravação são as de maior importância. Para além destas, poderia investir-se em outras, como por exemplo: fazer com que o BPM da aplicação seja variável de modo a fazer músicas que tenham um andamento maior ou menor consoante a intenção do utilizador; gravar durante mais tempos, em vez de apenas quatro, por exemplo 8, 12 e 16; criar um modo avançado em que possam ser dadas notas que não sejam apenas múltiplos de dois, como agora sucede, mas onde fosse possível dar três notas num só tempo, o que permitiria a construção de músicas com uma vertente mais *jazz*.

Relativamente à interação com a *Leap Motion,* a nova versão da *driver* (JavaScript SDK Documentation), lançada mais recentemente, inclui funcionalidades que permitem simplificar algumas das opções escolhidas para a interação. É possível agora quantificar o fechamento da mão do utilizador. Com esta funcionalidade, poderíamos eliminar a zona mais alta associada à mão esquerda e usar o fechamento da mão ou a sua abertura para saber se o utilizador pretende tocar ou não determinado som.

Ainda no domínio da interação, e tendo em conta que a *API* da *Leap Motion* está em constante mudança, será preciso estar atento às funcionalidades das novas versões para equacionar se a implementação de novos mecanismos de interação faz sentido na aplicação.

Finalmente, quanto à visualização e à cena tridimensional, podiam ser exploradas outras formas de visualização das várias *tracks* e tentar aproximar esta aplicação do modelo usado pela aplicação *A dive in Music* referido anteriormente. Nessa aplicação são apresentadas diversas visualizações que vão mudando aleatoriamente ao longo do tempo. Poder-se-ia inclusivamente mudar a estrutura da aplicação de modo a que fosse possível a qualquer utilizador construir a sua própria visualização e inclui-la na aplicação existente, formando-se assim um repositório de visualizações que pudessem

ser usadas e partilhadas por qualquer pessoa. Esta funcionalidade teria possuiria também uma vertente social e de partilha de visualizações entre os vários utilizadores.

Bibliografia

(s.d.).

(s.d.). Obtido em 2013 de December de 2013, de Leap Motion: https://www.leapmotion.com/

(s.d.). Obtido de http://glsl.heroku.com/e#15802.0

Ólafur Arnalds - Ljósið (Official Music Video). (s.d.). Obtido em 20 de December de 2013, de Vimeo: http://vimeo.com/6284199

3D Waveform. (s.d.). Obtido em 20 de December de 2013, de mrdoob: http://www.mrdoob.com/#/98/3d_waveform

A dive in Music. (s.d.). Obtido em 20 de December de 2013, de http://do.adive.in/music/

Ableton. (s.d.). Obtido em 20 de December de 2013, de Ableton: https://www.ableton.com/

After Effects. (s.d.). Obtido em 20 de December de 2013, de Adobe: http://www.adobe.com/products/aftereffects.html

audio keys. (s.d.). Obtido em 20 de December de 2013, de github: https://github.com/zz85/audiokeys.js

Audio Tool. (s.d.). Obtido em 20 de December de 2013, de Audio Tool: http://www.audiotool.com/

Autodesk. (s.d.). Obtido em 20 de December de 2013, de Autodesk: http://www.autodesk.com/

Beatbox brilliance: Tom Thum at TEDxSydney. (s.d.). Obtido de Youtube: https://www.youtube.com/watch?v=GNZBSZD16cY

Beatbox brilliance: Tom Thum at TEDxSydney. (8 de July de 2013). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=GNZBSZD16cY&feature=youtu.be&t=8m12s

BeatPetite. (s.d.). Obtido em 20 de December de 2013, de BeatPetite: http://beatpetite.com/

Bernhoft. (s.d.). Obtido em 20 de December de 2013, de Bernhoft: http://bernhoft.org/

Bernhoft - C'mon Talk (Official Video). (4 de November de 2011). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=rxoiZZ8UBEY

Bio. (s.d.). Obtido em 20 de December de 2013, de Zoe Keating: http://www.zoekeating.com/bio.html

cubevis. (s.d.). Obtido em 20 de December de 2013, de shoffing: http://shoffing.com/pages/projects/cubevis/

Dub FX 10/10/2008 'Love Someone'. (16 de October de 2008). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=UiInBOVHpO8

Dubbing into Dub FX. (s.d.). Obtido em 20 de December de 2013, de The Hindu: http://www.thehindu.com/todays-paper/tp-features/tp-metroplus/article913786.ece

DUBFX. (s.d.). Obtido em 20 de December de 2013, de DUBFX: http://dubfx.net/

DubFX - Love Someone. (s.d.). Obtido de Youtube: https://www.youtube.com/watch?v=UiInBOVHpO8

Erased Tapes Records. (s.d.). Obtido em 20 de December de 2013, de Erased Tapes Records: http://www.erasedtapes.com/

Esteban Diacono. (s.d.). Obtido em 20 de December de 2013, de Esteban Diacono: http://portfolio.estebandiacono.tv/

Flash. (s.d.). Obtido em 20 de December de 2013, de Abobe: http://www.adobe.com/products/flash.html

Fragment Shader. (s.d.). Obtido de OpenGl: https://www.opengl.org/wiki/Fragment_Shader

GARCIA, C. M. (2011). *STUDY ON THE PROCEDURAL GENERATION OF VISUALIZATION*. Texas.

Jarle Bernhoft Artist Biography by Jason Birchmeier. (s.d.). Obtido em 20 de December de 2013, de ALL MUSIC: http://www.allmusic.com/artist/jarle-bernhoft-mn0001474283/biography

JavaScript SDK Documentation. (s.d.). Obtido de Leap Motion: https://developer.leapmotion.com/documentation/skeletal/javascript/index.html

Leap Motion Structured Light Pattern. (18 de January de 2013). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=UI5EBzU_QqM

Loop Waveform Visualizer. (s.d.). Obtido em 20 de December de 2013, de Air Tight Interactive: http://airtightinteractive.com/demos/js/reactive/

Loop Waveform Visualizer. (s.d.). Obtido em 20 de December de 2013, de Airtight Interactive: http://www.airtightinteractive.com/2012/01/loop-waveform-visualizer/

Maya. (s.d.). Obtido em 20 de December de 2013, de Autodesk: http://www.autodesk.com/products/autodesk-maya/overview

Mozilla. (s.d.). *Using web workers*. Obtido de https://developer.mozilla.org/en/docs/Web/Guide/Performance/Using_web_workers

Mozilla. (s.d.). *WebGL*. Obtido de Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/WebGL

Music Colour Particles. (s.d.). Obtido em 20 de December de 2013, de jabtunes: http://jabtunes.com/labs/arabesque/

OpenGL Shading Language. (s.d.). Obtido de OpenGL: https://www.opengl.org/documentation/glsl/

Pringle, P. (10 de January de 2009). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=K6KbEnGnymk

Seungki Kim, W. L. (2013). *SOUND BOUND: Making a Graphic Equalizer More Interactive and Fun.* Paris.

SOFTSTEP. (s.d.). Obtido em 20 de December de 2013, de Keithmcmillen: http://www.keithmcmillen.com/softstep/overview

Sooperlooper. (s.d.). Obtido em 20 de December de 2013, de Essej: http://essej.net/sooperlooper/

spark.js. (s.d.). Obtido em 20 de December de 2013, de github: https://github.com/zz85/sparks.js/

Ssergejewitsch, T. L. (1928). *Patente N.º US1658953 A*. USA.

Stuart Memo. (s.d.). Obtido em 20 de December de 2013, de Stuart Memo: http://stuartmemo.com/

three.js. (s.d.). Obtido em 20 de December de 2013, de three.js: http://threejs.org/

Three.js. (s.d.). Obtido de Three.js: http://threejs.org/

Tone Craft. (s.d.). Obtido em 20 de December de 2013, de Tone Craft: http://labs.dinahmoe.com/#tonecraft

ToneMatrix. (s.d.). Obtido em 20 de December de 2013, de AudioTool: http://tonematrix.audiotool.com/

Trapcode Particular 2.2. (s.d.). Obtido em 20 de December de 2013, de Red Giant: http://www.redgiant.com/products/all/trapcode-particular/

Vertex Shader. (s.d.). Obtido de OpenGL: https://www.opengl.org/wiki/Vertex_Shader

Web Audio API. (s.d.). Obtido em 20 de December de 2013, de https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html#AnalyserNode

Web Audio API - Analyser Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-analysernode-interface

Web Audio API - Biquad Filter Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-biquadfilternode-interface

Web Audio API - Destination Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-audiodestinationnode-interface

Web Audio API - Dynamics Compressor. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-dynamicscompressornode-interface

Web Audio API - Gain Node. (s.d.). Obtido de Web Audio API : http://webaudio.github.io/web-audio-api/#the-gainnode-interface

Web Audio API - Panner Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-pannernode-interface

Web Audio API - Script Processor Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-scriptprocessornode-interface---deprecated

Web Audio API - Source Node. (s.d.). Obtido de Web Audio API: http://webaudio.github.io/web-audio-api/#the-audiobuffersourcenode-interface

Zoe Keating - Lost. (s.d.). Obtido de Youtube: https://www.youtube.com/watch?v=sG9H5E2JN3s

Zoe Keating live - 'Lost' [HD] Sound Quality, ABC Radio National. (2 de July de 2012). Obtido em 20 de December de 2013, de Youtube: http://www.youtube.com/watch?v=sG9H5E2JN3s

Anexo A : Guião de Testes com Utilizadores

Esta é uma aplicação que tem como objectivo fazer música. Existem vários sons que pode tocar, que se encontram na zona inferior do ecrã. Ao selecionar um som pode tocá-lo utilizando o dispositivo Leap Motion que se encontra à sua frente. A mão esquerda serve para controlar o momento em que o som é tocado e a mão direita é utilizada para controlar o som.

Pode gravar o som que selecionou ao carregar no botão circular que se encontra junto do som selecionado. Quando acabar de gravar, o som é automaticamente reproduzido. Depois disto pode repetir o processo com outro som diferente de modo a criar música. Além de poder criar som, também pode visualizá-lo em 3D.

Por favor experimente a aplicação e de seguida responda às seguintes perguntas.

Idade : ☐

Sexo : M☐ F☐

Costuma jogar jogos de computador/consolas? Sim☐ Não☐

Sabe tocar algum instrumento musical? Sim☐ Não☐

☐ Canhoto ☐ Destro ☐ Ambidestro

Áudio

1 - O que acha da escolha dos sons? Resultam bem quando tocados em conjunto?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Não resultam bem	Resultam mal	Satisfatório	Resultam Bem	Ideal

2 - Sente que é uma vantagem ter o som da batida a tocar desde o início? Ou preferia que a aplicação começasse em silêncio e controlar o utilizador o som da batida?

☐ ☐

Com batida automática Em silêncio, controlar a batida

3 – A velocidade da batida - e consequentemente do resto dos sons - é ideal? Deveria ser mais rápida? Mais lenta?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Muito mais lenta	Mais lenta	Ideal	Mais rápido	Muito mais rápido

4 – Os sons que nesta fase estão presentes na aplicação são apenas de ritmo. Gostaria de poder tocar sons com notas?

☐ ☐

Sim Não

5 – O processo de reprodução e gravação é fácil de utilizar?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Muito difícil de utilizar	Difícil de utilizar	Satisfatório	Fácil de utilizar	Muito fácil de utilizar

Interação

1 – A mão esquerda pode tocar 4 zonas diferentes, cada uma destas zonas toca o som em momentos diferentes. Preferia ter mais zonas e assim poder ter mais escolhas quanto ao tocar o som mas, ao mesmo tempo, ter menos precisão ao situar a mão numa zona? Ou preferia o oposto, menos zonas e mais precisão?

☐ 4 zonas é ideal

☐ Outro número : ☐

2 – Consegue identificar claramente a zona em que as suas mãos se encontram?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Identificação impossível	Identificação difícil	Com ligeira dificuldade	Identificação fácil	Claramente identificável

3 – O que acha das cores que são usadas para delimitar as zonas das mãos do utilizador? (cinzento e verde)

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Muito má escolha	Má escolha	Escolha pertinente	Boa escolha	Muito boa escolha

4 – E relativamente às escolhas das cores das mãos? (azul e vermelho)

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Muito má escolha	Má escolha	Escolha pertinente	Boa escolha	Muito boa escolha

5 – Prefere controlar o volume com a mão direita ou preferiria controlar outro parâmetro? Por exemplo algum efeito no som, como distorção?

☐	☐
Volume	Outro parâmetro

6 – Tendo em conta que é destro/canhoto gostaria que as funcionalidades das mãos estivessem trocadas? Ou seja, volume na mão esquerda e momento em que o som é tocado na mão direita?

☐	☐
Como está	Trocado

Visualização

1 – A visualização de cada som é claramente identificável?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Identificação impossível	Identificação difícil	Com ligeira dificuldade	Identificação fácil	Claramente identificável

2 – Tendo em conta o som base da batida, acha que a sua visualização é adequada?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Nada adequada	Pouco adequada	Satisfatória	Muito adequada	Ideal

3 – O que acha da cena no seu conjunto?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Não resulta bem	Resulta mal	Satisfatório	Resulta Bem	Resulta muito bem

4 – Parece-lhe que as 5 visualizações dos blocos de paralelepípedos foram boas escolhas? Tem alguma sugestão para outra visualização?

1 ☐	2 ☐	3 ☐	4 ☐	5 ☐
Muito má escolha	Má escolha	Escolha pertinente	Boa escolha	Muito boa escolha

Sugestão:

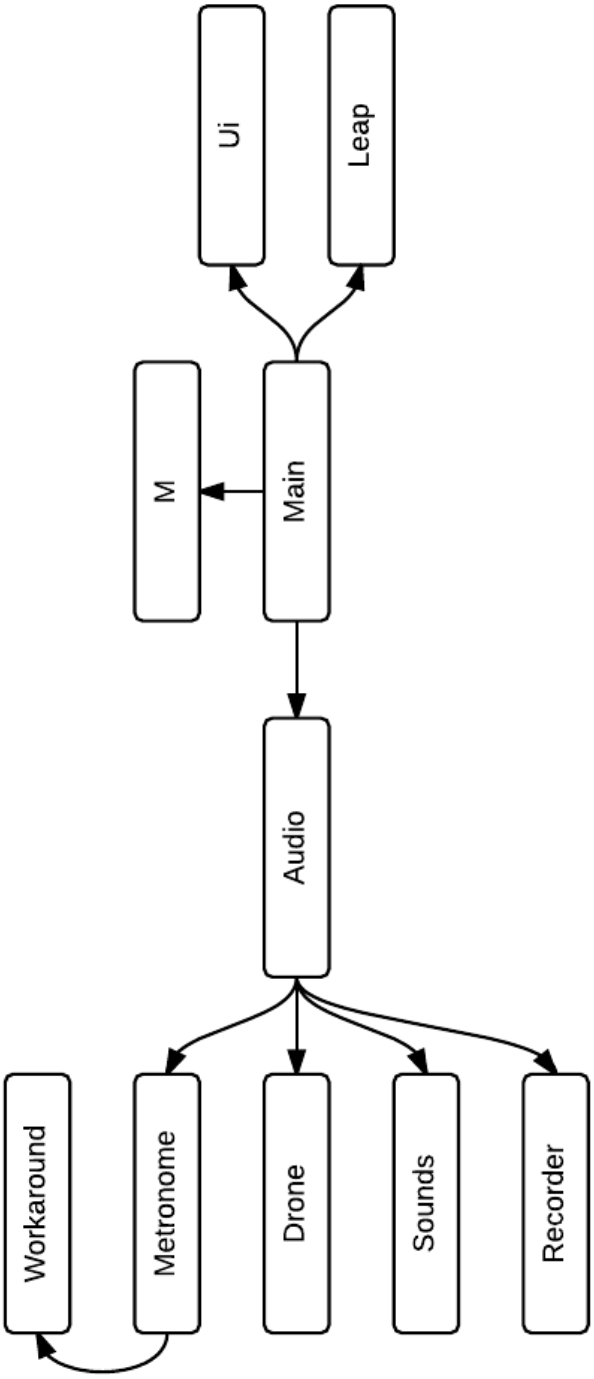
5 – O que acha da escolha das cores dos vários conjuntos de paralelepípedos? São demasiado parecidas? Cada um deveria ter uma cor própria que o distinguisse dos outros?

☐ Está bem como está

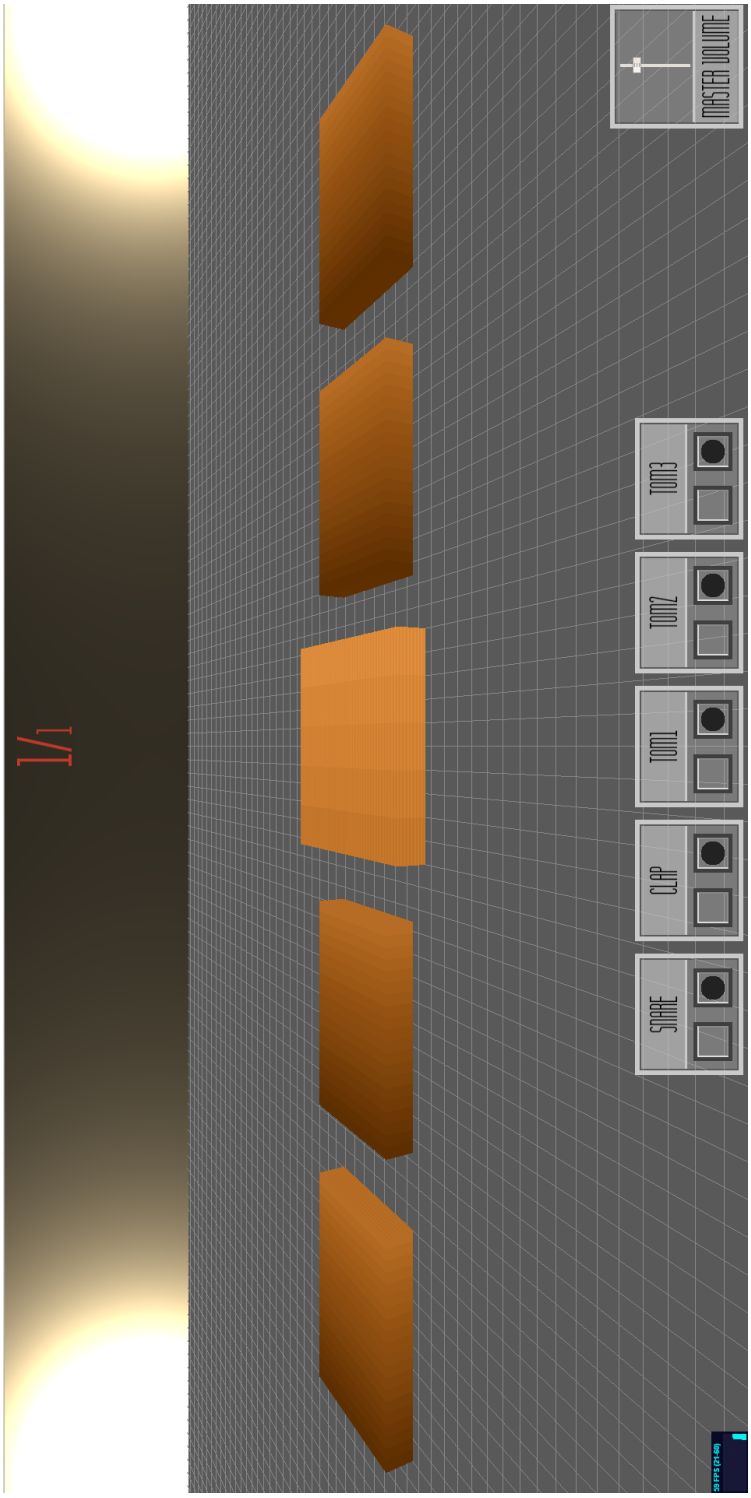
☐ Cores individuais

Comentários / Observações / Problemas

Anexo B : Diagrama Oraganizacional da Aplicação

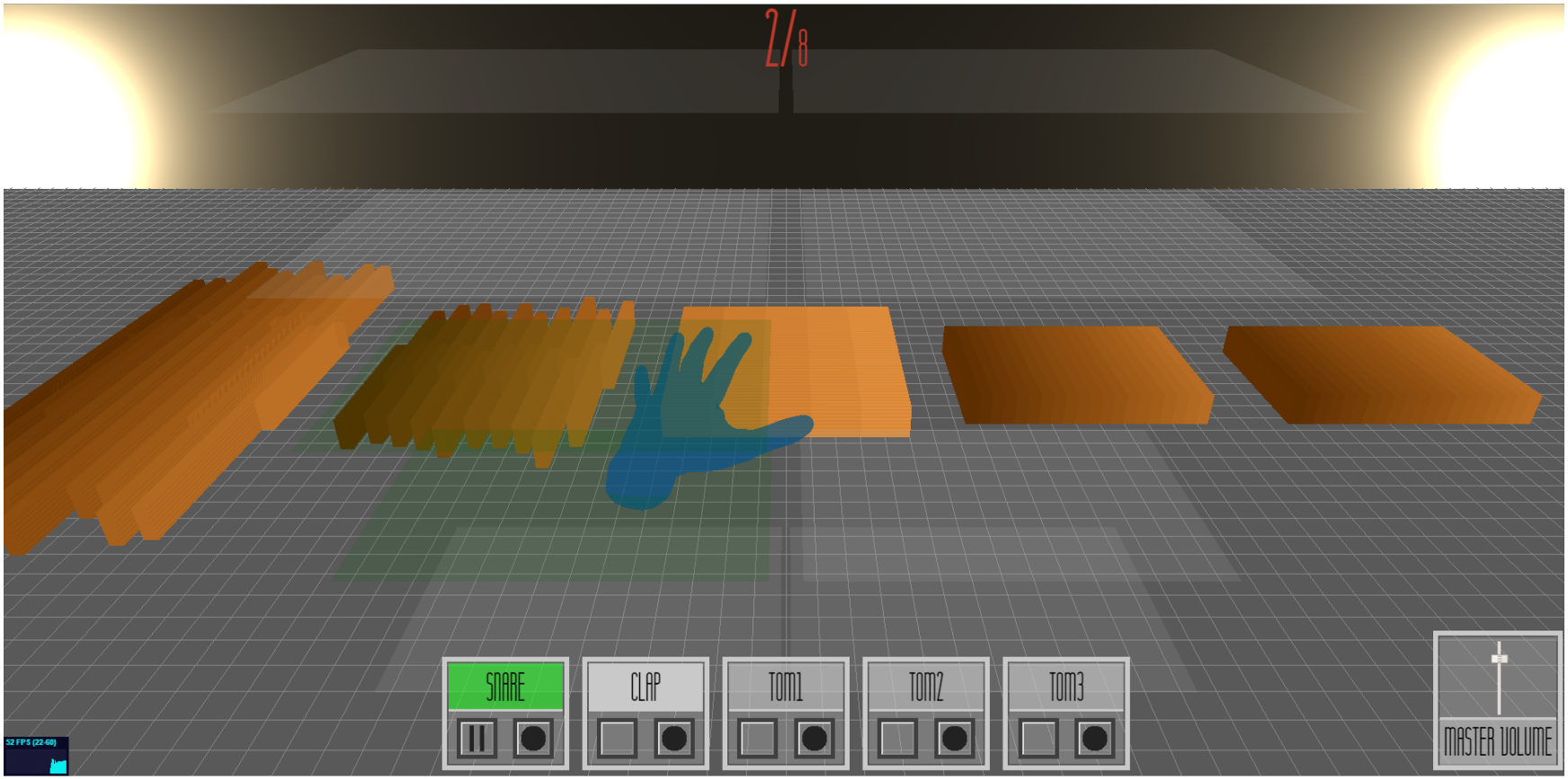


Anexo C : Capturas de Ecrã



3/12
SNARE
CLAP
TOM1
TOM2
TOM3
MASTER VOLUME

4/14
SNARE
CLAP
TOM1
TOM2
TOM3
MASTER VOLUME


2/8
SNARE
CLAP
TOM1
TOM2
TOM3
MASTER VOLUME